Universidade Federal do Paraná

Olacir Rodrigues Castro Junior

Bio-inspired optimization algorithms for multi-objective problems

Curitiba PR
2017

Olacir Rodrigues Castro Junior

Bio-inspired optimization algorithms for multi-objective problems

Thesis presented as partial fulfillment of the requirements for the attainment of the degree of Doctor in Informatics in the Graduate Program in Informatics, Department of Exact Sciences of the Federal University of Paraná.

Field: *Computer Science*.

Advisor: Aurora Trinidad Ramirez Pozo.

Co-advisor: Roberto Santana Hermida.

Curitiba PR
2017

C355b Castro Junior, Olacir Rodrigues
Bio-inspired optimization algorithms for multi-objective problems / Olacir Rodrigues Castro Junior. – Curitiba, 2017.
172 f ; il. color : 30 cm.

Tese - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática, 2017.

Orientador: Aurora Trinidad Ramirez Pozo – Co-orientador: Roberto Santana Hermida
Bibliografia: p. 161-172.

1. Algorítmos. 2. Algoritmos computacionais. 3. Programação heurística. 4. Programação (Matemática) . I. Universidade Federal do Paraná. II.Pozo, Aurora Trinidad Ramirez. III. Hermida, Roberto Santana . IV. Título.

CDD: 005.1

PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
Setor CIÊNCIAS EXATAS
Programa de Pós-Graduação INFORMÁTICA

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da tese de Doutorado de **OLACIR RODRIGUES CASTRO JUNIOR** intitulada: **Bio-inspired optimization algorithms for multi-objective problems**, após terem inquirido o aluno e realizado a avaliação do trabalho, são de parecer pela sua Aprovação.

Curitiba, 06 de Março de 2017.

AURORA TRINIDAD RAMIREZ POZO
Presidente da Banca Examinadora (UFPR)

CELSO YOSHIKAZU ISHIDA
Avaliador Externo (UFPR)

JOSE ANTONIO LOZANO
Avaliador Externo (UPV/EHU)

ROBERTO SANTANA HERMIDA
Co-orientador - Avaliador Externo (UPV/EHU)

LEANDRO DOS SANTOS COELHO
Avaliador Externo (PUC/PR)

MYRIAM REGATTIERI DE BIASE DA SILVA DELGADO
Avaliador Externo (UTFPR)

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ
Setor de Ciências Exatas
Coordenação de Pós-Graduação em Informática

Contents

List of Figures

List of Tables

List of Abbreviations

ACF	Adaptive Choice Function
AE	Algorithm Effort
AMALGAM	Multialgorithm, Genetically Adaptive Multiobjective
AMEDA	Adaptive Multi-objective Estimation of Distribution Algorithms
ANOVA	Analysis Of Variance
AOS	Adaptive Operator Selection
BIC	Bayesian Information Criterion
BOA	Bayesian Optimization Algorithm
C-Multi	Competent Multi-swarm
CD	Crowding Distance
CEC	IEEE Congress on Evolutionary Computation
CF	Choice Function
CMA-ES	Covariance Matrix Adaptation Evolution Strategy
COCO	Comparing Continuous Optimizers
CPSO	Co-operative Particle Swarm Optimization
CPSO-S	Co-operative split Particle Swarm Optimization
CVEPSO	Co-operative Vector-evaluated Particle Swarm Optimization
DB	Davies-Bouldin index
DE	Differential Evolution
DTLZ	Deb, Thiele, Laumanns, Zitzler
EA	Evolutionary Algorithm
EDA	Estimation of Distribution Algorithm
EHU	Euskal Herriko Unibertsitatea
EvE	Exploration vs. Exploitation
FIR	Fitness Improvement Rate
FRRMAB	Fitness-Rate-Rank-based Multi-Armed Bandit
FWER	Family-Wise Error Rate
GA	Genetic Algorithm
GD	Generational Distance
GNG	Growing Neural Gas network
H-MOPSO	Hyper-heuristic Multi-objective Particle Swarm Optimization
HH_CF	Hyper-Heuristic based on Choice Function
HMOEA	Hybrid Multi-objective Evolutionary Algorithm
HV	Hypervolume
I-DBEA	Improved Decomposition-Based Evolutionary Algorithm
I-Multi	Iterated Multi-swarm
IBEA	Indicator Based Evolutionary Algorithm
IE	Improving and Equal
IGD	Inverted Generational Distance

KnEA	Knee point-driven Evolutionary Algorithm
MAB	Multi-Armed Bandit
MaOP	Many-objective Problem
MB-GNG	Growing Neural Gas Network designed for Model Building
MGA	Multilevel Grid Archiving
MGVEPSO$_A$	Multi Guided VEPSO with random archive selection
MO-CMA-ES	Multi-objective CMA-ES
MOBBO	Multi-objective Biogeography Based Optimization
MOEA	Multi-objective Evolutionary Algorithm
MOEA/D	Multi-objective Evolutionary Algorithm based on Decomposition
MOEA/D-CMA	MOEA/D Covariance Matrix Adaptation
MOEA/D-DRA	MOEA/D with Dynamic Resource Allocation
MOEA/D-HH	MOEA/D Hyper Heuristic
MOEA/DD	Multi-objective Evolutionary Algorithm based on Dominance and Decomposition
MOEDA	Multi-objective Estimation of Distribution Algorithm
MOGA	Multi Objective Genetic Algorithm
MONEDA	Multi-objective Optimization with a Neural Network-based EDA
MOP	Multi-objective Problems
MOPSO	Multi-objective Particle Swarm Optimization
MOPSO/D	Multi-objective Particle Swarm Optimization based on Decomposition
NSGA	Non-dominated Sorting Genetic Algorithm
NWSum	New WSum
OMOPSO	Optimized MOPSO
PBI	Penalty-based Boundary Intersection
PDF	Probability Density Function
PF	Pareto Front
PLS	Probabilistic Logic Sampling
PM	Probabilistic Modeling
PS	Pareto Set
PSO	Particle Swarm Optimization
R-Multi	Reference-Point Based Multi-swarm
rBOA	real-coded Bayesian Optimization Algorithm
RNI	Ratio of Non-dominated Individuals
SBX	Simulated Binary Crossover
SMPSO	Speed-constrained Multi-objective Particle Swarm Optimization
SMS-EMOA	S-Metric Selection Evolutionary Multi-objective Algorithm
SPEA	Strength Pareto Evolutionary Algorithm
TCH	Tchebycheff
TWF	Transfer Weight Function
UB	Unbounded
UCB	Upper Confidence Bound
UD	Uniform Distribution
UP-MO-CMA-ES	Unbounded Population MO-CMA-ES
UPV	Universidad del País Vasco
VEPSO	Vector-evaluated Particle Swarm Optimization
WFG	Walking Fish Group
WSum	Weighted Sum

List of Symbols

n	number of decision variables of an optimization problem
Ω	feasible space of decision variables
$\boldsymbol{x} = (x_1, ..., x_n)$	n-dimensional decision variable vector
$f(\boldsymbol{x})$	an objective function of a single-objective problem
m	number of objective functions of a multi-objective problem
$\boldsymbol{f}(\boldsymbol{x}) = (\boldsymbol{f}_1(\boldsymbol{x}), ..., \boldsymbol{f}_m(\boldsymbol{x}))$	m-dimensional objective function vector
Λ	attainable space of objectives
$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{u} = (u_1, ..., u_m)$	m-dimensional objective function vector
$\mathcal{P}^*$	Pareto optimal set, contains all non-dominated solutions regarding a decision space
$\mathcal{PF}^*$	Pareto front, the image of $\mathcal{P}^*$ in the objective space
PF_t	true Pareto front, known mathematically
PF_k	approximate Pareto front obtained by an optimizer (known by the optimizer)
$N = \lvert\mathcal{P}^*\rvert = \lvert\mathcal{PF}^*\rvert$	number of solutions contained in the Pareto set, and consequently in the Pareto front
d	distance-related parameters for WFG problems
p	position related parameters for WFG and DTLZ problems
$\boldsymbol{r}$	the nadir (anti-optimal or “worst possible") point in space
$\boldsymbol{z}^*$	the ideal (optimal or “best possible") point in space
$\boldsymbol{w} = (w_1, ..., w_m), \sum_{i=1}^{m} w_i = 1$	m-dimensional weight vector
$\boldsymbol{W}$	set of uniformly distributed weight vectors
$\boldsymbol{v} = (v_1, ..., v_n)$	n-dimensional velocity vector in PSO
t	the current iteration of an optimizer
t_{max}	maximum number of iterations
$\boldsymbol{x}_{b_i}$	the best solution found by solution $\boldsymbol{x}_i$ (its local best) in PSO
$\boldsymbol{x}_{g_i}$	the best solution found by the neighborhood of $\boldsymbol{x}_i$ (its global best) in PSO
ω	the inertia weight in PSO
C_1	parameter that controls the influence of the personal best leader in PSO
C_2	parameter that controls the influence of the global best leader in PSO
χ	constriction factor for the SMPSO
$\boldsymbol{ub} = (ub_1, ..., ub_n)$	n-dimensional vector representing the upper bounds of the decision variables

$\boldsymbol{lb} = (lb_1, ..., lb_n)$	n-dimensional vector representing the lower bounds of the decision variables
T	neighborhood size in MOEA/D
δ	probability of using the neighborhood in selection/replacement of MOEA/D
$\boldsymbol{P}$	current population of an optimizer
$\overline{\boldsymbol{P}}$	set of the best individuals selected from a population
$\boldsymbol{O}$	set of offspring solutions
$\mathcal{M}$	a probabilistic model
ζ	the structure of a probabilistic model
$\boldsymbol{\theta}$	the parameters of a probabilistic model
$\boldsymbol{X} = (X_1, ..., X_n)$	n-dimensional vector of random variables, whose instantiations are $\boldsymbol{x} = (x_1, ..., x_n)$
$\boldsymbol{\Pi}_{\boldsymbol{X}_i}$	the set of parents of X_i
BIC_λ	the complexity factor of the BIC score in rBOA
$\bar{\boldsymbol{m}}$	the mean of a CMA-ES distribution
$\boldsymbol{C}$	the covariance matrix of CMA-ES
$\boldsymbol{B}$	eigenvectors on a eigendecomposition of $\boldsymbol{C}$
$\boldsymbol{D}$	eigenvalues on a eigendecomposition of $\boldsymbol{C}$
σ	the step size of CMA-ES
$\boldsymbol{p_\sigma}$	the evolution path for updating the step size of CMA-ES
$\boldsymbol{p_c}$	the evolution path for updating the covariance matrix of CMA-ES
λ	number of solutions sampled in CMA-ES
μ	number of selected solutions
$\mathcal{N}(\bar{\boldsymbol{m}}, \boldsymbol{C})$	multivariate normal distribution with mean $\bar{\boldsymbol{m}}$ and covariance matrix $\boldsymbol{C}$
M_k	parameter of the Minkowski metric that defines a family of metrics

Resumo

Problemas multi-objetivo (MOPs) são caracterizados por terem duas ou mais funções objetivo a serem otimizadas simultaneamente. Nestes problemas, a meta é encontrar um conjunto de soluções não-dominadas geralmente chamado conjunto ótimo de Pareto cuja imagem no espaço de objetivos é chamada frente de Pareto. MOPs que apresentam mais de três funções objetivo a serem otimizadas são conhecidos como problemas com muitos objetivos (MaOPs) e vários estudos indicam que a capacidade de busca de algoritmos baseados em Pareto é severamente deteriorada nesses problemas. O desenvolvimento de otimizadores bio-inspirados para enfrentar MOPs e MaOPs é uma área que vem ganhando atenção na comunidade, no entanto, existem muitas oportunidades para inovar. O algoritmo de enxames de partículas multi-objetivo (MOPSO) é um dos algoritmos bio-inspirados adequados para ser modificado e melhorado, principalmente devido à sua simplicidade, flexibilidade e bons resultados. Para melhorar a capacidade de busca de MOPSOs, seguimos duas linhas de pesquisa diferentes: A primeira foca em métodos de líder e arquivamento. Trabalhos anteriores apontaram que esses componentes podem influenciar no desempenho do algoritmo, porém a seleção desses componentes pode ser dependente do problema. Uma alternativa para selecioná-los dinamicamente é empregando hiper-heurísticas. Ao combinar hiper-heurísticas e MOPSO, desenvolvemos um novo *framework* chamado H-MOPSO. A segunda linha de pesquisa também é baseada em trabalhos anteriores do grupo que focam em múltiplos enxames. Isso é feito selecionando como base o *framework* multi-enxame iterado (I-Multi), cujo procedimento de busca pode ser dividido em busca de diversidade e busca com múltiplos enxames, e a última usa agrupamento para dividir um enxame em vários sub-enxames. Para melhorar o desempenho do I-Multi, exploramos duas possibilidades: a primeira foi investigar o efeito de diferentes características do mecanismo de agrupamento do I-Multi. A segunda foi investigar alternativas para melhorar a convergência de cada sub-enxame, como hibridizá-lo com um algoritmo de estimativa de distribuição (EDA). Este trabalho com EDA aumentou nosso interesse nesta abordagem, portanto seguimos outra linha de pesquisa, investigando alternativas para criar versões multi-objetivo de um dos EDAs mais poderosos da literatura, chamado estratégia de evolução baseada na adaptação da matriz de covariância (CMA-ES). Para validar o nosso trabalho, vários estudos empíricos foram conduzidos para investigar a capacidade de busca das abordagens propostas. Em todos os estudos, nossos algoritmos investigados alcançaram resultados competitivos ou melhores do que algoritmos bem estabelecidos da literatura.

Palavras-chave: multi-objetivo, algoritmo de estimativa de distribuição, otimização por enxame de partículas, multiplos enxames, híper-heuristicas.

Abstract

Multi-Objective Problems (MOPs) are characterized by having two or more objective functions to be simultaneously optimized. In these problems, the goal is to find a set of non-dominated solutions usually called Pareto optimal set whose image in the objective space is called Pareto front. MOPs presenting more than three objective functions to be optimized are known as Many-Objective Problems (MaOPs) and several studies indicate that the search ability of Pareto-based algorithms is severely deteriorated in such problems. The development of bio-inspired optimizers to tackle MOPs and MaOPs is a field that has been gaining attention in the community, however there are many opportunities to innovate. Multi-objective Particle Swarm Optimization (MOPSO) is one of the bio-inspired algorithms suitable to be modified and improved, mostly due to its simplicity, flexibility and good results. To enhance the search ability of MOPSOs, we followed two different research lines: The first focus on leader and archiving methods. Previous works have pointed that these components can influence the algorithm performance, however the selection of these components can be problem-dependent. An alternative to dynamically select them is by employing hyper-heuristics. By combining hyper-heuristics and MOPSO, we developed a new framework called H-MOPSO. The second research line, is also based on previous works of the group that focus on multi-swarm. This is done by selecting as base framework the iterated multi swarm (I-Multi) algorithm, whose search procedure can be divided into diversity and multi-swarm searches, and the latter employs clustering to split a swarm into several sub-swarms. In order to improve the performance of I-Multi, we explored two possibilities: the first was to further investigate the effect of different characteristics of the clustering mechanism of I-Multi. The second was to investigate alternatives to improve the convergence of each sub-swarm, like hybridizing it to an Estimation of Distribution Algorithm (EDA). This work on EDA increased our interest in this approach, hence we followed another research line by investigating alternatives to create multi-objective versions of one of the most powerful EDAs from the literature, the Covariance Matrix Adaptation Evolution Strategy (CMA-ES). In order to validate our work, several empirical studies were conducted to investigate the search ability of the approaches proposed. In all studies, our investigated algorithms have reached competitive or better results than well established algorithms from the literature.

Keywords: multi-objective, estimation of distribution algorithms, particle swarm optimization, multi-swarm, hyper-heuristics.

Chapter 1

Introduction

Multi-Objective Problems (MOPs) arise naturally in most disciplines. This type of problem is characterized by having two or more objective functions to be optimized simultaneously. Thus, there is no unique optimal solution, but a set of them. These solutions are usually found through the use of Pareto optimality theory, hence, these solutions are known as Pareto optimal set and its image in the objective space is known as Pareto front (Coello et al., 2006).

A good optimizer for this kind of problem must provide a set of solutions that is both: close to the optimal Pareto front (convergence) and well spread along it (diversity). However, these two goals are usually conflicting, therefore, the main challenge of an optimizer is to enhance both.

For MOPs having two or three objectives, the Pareto dominance relation is usually enough for guiding an optimizer, however in problems that involve more than three objectives, known as Many-Objective Problems (MaOPs), the search ability of Pareto-based algorithms is severely deteriorated (Britto and Pozo, 2012; Deb and Jain, 2014; Ishibuchi et al., 2008; Nebro et al., 2009).

Currently, many researchers are proposing bio-inspired optimizers to deal with MOPs and MaOPs (Deb and Jain, 2014; Krause et al., 2016; Li et al., 2015). In this context, developing efficient algorithms can be a difficult task due to the great amount of competition with state-of-the-art approaches. However, since it is a challenging area, there are still many opportunities to innovate.

Bio-inspired algorithms based on many different ideas have been devised to deal with these challenges, like coral (Salcedo-Sanz et al., 2015), bacteria (Niu et al., 2013), bees (Luo et al., 2017), fish schools (Bastos-Filho and Guimarães, 2015), among others (Bechikh et al., 2017), we need to select a suitable research line. In order to begin our work, we selected as basic framework, the Multi-Objective Particle Swarm Optimization (MOPSO) algorithm. MOPSO belongs to the class of swarm intelligence algorithms, however, it has similarities to Multi-objective Evolutionary Algorithms (MOEAs), like dealing with several solutions simultaneously and seeking to improve the quality of these solutions along the iterations. Moreover, due to these similarities, several ideas employed in this work with MOPSOs were firstly proposed for evolutionary computation. The MOPSO framework was selected as our basic algorithm due to its good results (Nebro et al., 2009), simplicity, and flexibility. Further, our research group has been working with MOPSO for some time and is familiarized with it.

In previous works, our research group already investigated two important components of MOPSOs: the leader selection (Castro Jr. et al., 2012) and archiving methods (Britto and Pozo, 2012). Although we had identified that these components influence the performance of

MOPSOs, selecting the proper algorithm components to optimize a MOP is a hard task because they are problem-dependent.

Among the alternatives to deal with this issue, we can highlight the use of hyper-heuristics. By combining hyper-heuristics with MOPSO, we developed a new framework called Hyper-heuristic Multi-objective Particle Swarm Optimization (H-MOPSO), which considers the leader and archiving methods as low-level heuristics, and can employ any heuristic selection mechanism to dynamically determine the best components during the optimization.

In parallel, other line of work that our research group has been working within MOPSOs, is the use of multiple swarms. Mainly we are concentrated in the iterated multi swarm (I-Multi) (Britto et al., 2013) algorithm. The search procedure of I-Multi can be divided in two phases: diversity and multi-swarm searches. In the first phase, the goal is to find a well diversified set of solutions. In the second phase, several well-distributed sub-swarms are created and each one focuses on convergence. I-Multi uses a clustering mechanism to split an initial swarm into several sub-swarms.

In order to improve the search ability of I-Multi, two new possibilities can be explored: The first is to further investigate the effect of different characteristics of the clustering mechanism of I-Multi, like the space in which the clustering is performed and the similarity metric used for clustering.

The second is to investigate alternatives to improve the convergence of each sub-swarm of I-Multi. One of these alternatives is to hybridize with other powerful algorithms. Among different alternatives from the literature, we selected an estimation of distribution algorithm (EDA), mostly because EDAs usually present good convergence characteristics. Moreover, we developed a partnership with the University of the Basque Country (UPV/EHU) that could aid us in the investigation of EDAs.

Among several EDAs from the literature (Hauschild and Pelikan, 2011), we selected the real-coded Bayesian optimization algorithm (rBOA) (Ahn et al., 2006), which is a continuous version of the well-known Bayesian optimization algorithm (BOA) (Pelikan et al., 1999), one of the most efficient and extensively applied EDAs (Hauschild and Pelikan, 2011). Moreover, rBOA presents good results in the literature and has its source code available in the web page of the authors.

This work with EDAs increased our interest in this approach. Hence, we decided to follow another research line by investigating alternatives to create multi-objective versions of one of the most powerful EDAs from the literature: the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (Hansen and Ostermeier, 1996). Although several variants of CMA-ES exist for MOPs (Igel et al., 2007a,b; Krause et al., 2016; Voß et al., 2008; Zapotecas-Martínez et al., 2015), there are still many research alternatives available to be explored in this area.

In our thesis proposal, our goal was to study all these topics (multi-swarm, hyper-heuristic, EDAs) and combine them maximizing their strengths and minimizing their weaknesses. Possible combinations of these topics could include using hyper-heuristics to select between MOPSOs or EDAs to be used in each sub-swarm of I-Multi. However, we exhausted our available time before having a chance to further investigate combinations between these approaches, hence the investigation of alternatives to combine our different lines of research is due to future works.

1.1 Objectives

The objective of this work is to devise optimizers able to deal with MOPs and MaOPs.

To pursue this main goal we investigated three very distinct research lines that can be combined into powerful algorithms. The research lines investigated were: MOPSOs with multiple swarms, hyper-heuristics and EDAs.

1.2 Contributions

This work has generated mainly four contributions. The first of them was the identification and parameterization of an effective hyper-heuristic to dynamically select leader and archiving methods for a MOPSO algorithm. There are few works in the literature that apply hyper-heuristics on multi-objective optimizers, and this work, as far as we know, is the first using a hyper-heuristic to select leader and archiving methods in a MOPSO algorithm.

The second contribution was the investigation of two important components of clustering for multi-swarm MOPSOs. The space in which the clustering is performed and the similarity metric used to perform this clustering. As far as we know, this is the first work where the performance of different clustering strategies for MOPs is linked to the modalities of difficulty of the functions. Similarly, we have not found any previous study on the impact of the similarity metrics on the behavior of the algorithms.

The third contribution was the combination between two good optimizers from the literature: I-Multi and rBOA. Where we identified that rBOA is a good approach to intensify the convergence in the multi-swarm phase of I-Multi. The most innovative parts in this approach were the use of multiple rBOA models simultaneously and the hybridization with a MOPSO algorithm.

In our last contribution, we proposed three very distinct multi-objective CMA-ES (MO-CMA-ES) approaches. The first one is the least powerful, because it uses a single CMA-ES instance to model the entire search space. The second variant, as the first, was based on Pareto dominance, however, each individual encodes its own CMA-ES model. The third variant was created with MaOPs in mind, hence it is based on dominance and decomposition concepts simultaneously, moreover, each individual had its own CMA-ES model as well. As far as we know, this is the first work to present the idea of using Transfer Weight Functions (TWFs) as flexible components to manage the incorporation of information into a Pareto based MO-CMA-ES, moreover, we compared the performance of different TWFs. Also, in this study, for the first time, information from the closest solutions was used to update the model associated to a solution in a Pareto based framework. Moreover, for the first time the CMA-ES and MOEA/DD algorithms were combined.

The studies conducted during this work have produced papers published in prestigious journals (Castro Jr. et al., 2016a,b) and conferences (Castro Jr. and Pozo, 2014a,b, 2015). Moreover, there are still papers under review for journals (Castro Jr. et al., a,c) and one conference (Castro Jr. et al., b).

1.3 Organization

The remaining of this work is organized as follows:

Chapter 2 - Related work: this chapter briefly introduces some recently proposed work on the field of MOPSOs, MOEAs, Hyper-heuristics and EDAs.

Chapter 3 - Background: this chapter presents the main concepts used in this work, including single and multi-objective optimization problems, important optimization algorithms,

like MOPSO, MOEA/D and variants, two EDAs, three selection hyper-heuristics, as well as the benchmark problems, performance metrics and statistical tests used throughout this work.

Chapter 4 - Using hyper-heuristics to select leader and archiving methods for MOPSO: this chapter presents our investigation on the use of different heuristic selection methods and their parameterization to select leader and archiving methods for MOPSO.

Chapter 5 - Investigating clustering strategies on I-Multi: this chapter contains the investigation conducted on two important characteristics of clustering: the space in which the clustering is performed and the similarity metric used.

Chapter 6 - Hybrid competent multi-swarm based on rBOA and MOPSO: this chapter presents the studies conducted on the hybridization of I-Multi and rBOA.

Chapter 7 - CMA-ES approaches for multi objective problems: this chapter depicts the different approaches devised to create efficient multi-objective variants of the CMA-ES algorithm.

Chapter 8 - Conclusion: finally our conclusions are presented in this chapter and possible future works are discussed.

Chapter 2

Related work

In this thesis, we work mainly with three research lines to enhance the search ability of optimizers for solving MOPs and MaOPs: MOPSOs employing multiple swarms, hyper-heuristics and EDAs. Moreover, our starting point was the MOPSO framework, which we deeply modified along this work. Hence, in this chapter, we present recently proposed, representative approaches of our areas of study. The purpose of this chapter is not to make a comprehensive review of the literature, but only to contextualize the fields in which our work is inserted.

We start by presenting a few MOPSO variants in Section 2.1. Multi-swarm MOPSOs, employing different approaches for split and communication among the swarms are discussed in Section 2.2. Despite of being interested in MOPs, in this work we are deeply interested in finding efficient optimizers for solving MaOPs, Section 2.3 presents state-of-the-art MOEAs for optimizing this kind of problem. Multi-objective optimizers based on hyper-heuristics or Adaptive Operator Selection (AOS), since it is a closely related area, are presented on Section 2.4. Section 2.5 brings a few MOEDAs from the literature based on CMA-ES and other Probabilistic Modeling (PM) approaches. Finally a brief discussion of this chapter is given in Section 2.6.

2.1 MOPSOs

A MOPSO based on the hypervolume indicator is proposed by García et al. (2014) and called MOPSOhv. This algorithm uses the hypervolume contribution of the archived solutions both to select the individuals that will be considered as global and personal leaders for each particle, and as archiver, so the solutions with the highest hypervolume are kept. The results indicate that MOPSOhv is competitive to state-of-the-art MOEAs for few objectives, and outperforms them for many-objectives.

A MOPSO algorithm without the personal best is proposed by Wang and Xu (2014). The authors argue that the personal best in a MOPSO leads the particles to less optimal solutions and only brings randomness. If before the velocity update, the repository size is smaller than the population size, the usual algorithm is used. Otherwise, if the repository exceeds the population size, the algorithm runs without the personal best by simply setting 0 to the personal best acceleration coefficient. In general, best results were achieved using the new strategy that discards the personal best.

A MOPSO based on decomposition is proposed by Dai et al. (2015) and called Multi-objective Particle Swarm Optimization based on Decomposition (MOPSO/D). This algorithm is composed of four parts: space decomposition and population classification; update strategy; recombination operations and selection strategy. The space decomposition, decomposes the MOP into several sub-regions, each one defined by a direction vector and assigns a solution to

each sub-region. In the update strategy, if a sub-region has no solutions associated, the solution whose objective vector has the smallest angle to the direction vector of the sub-region is selected and kept. Otherwise, if the sub-region has only one solution, it is kept. If the subproblem has two or more solutions, only the non-dominated are considered, and the one whose objective vector has the smallest angle to the direction vector of the sub-region is selected and kept. Two recombination operations are used in MPSO/D: the recombination operations of the PSO and a neighborhood correction proposed by the authors and based on the personal best and global best solutions of the current particle. Two selection strategies are used in MPSO/D. The first selects the particle to be updated and is based on binary tournament based on the crowding distance. The second is to determine the personal best and global best of each selected particle. The personal best is randomly selected among the neighboring particles. Two strategies to select the global best are presented, the first randomly selects it from the population, the second selects the solution whose objective vector has the smallest angle to the center vector of the neighboring vectors. The results confirm that MPSO/D outperforms classical MOEAs in most of the investigated problems.

A MOPSO based on the R2 indicator is proposed by Díaz-Manríquez et al. (2016) and called R2-MOPSO. This MOPSO is based on the single objective PSO and changes the operator used to compare the solutions to the contributing R2. Moreover, an external archive is used only to store the best non-dominated solutions found so far, the contributing R2 is used as archiver. The results indicate that R2-MOPSO is competitive with state-of-the-art approaches for two and three objectives. In a many-objective scenario, R2-MOPSO outperformed the well-known SMS-EMOA (Beume et al., 2007) using approximate hypervolume.

Among the MOPSOs presented here, the MOPSOhv is promising, however its applicability is limited to problems having few objectives, due to the high computational cost of calculating the hypervolume. Alternatively the approximate hypervolume or other indicator like R2 could be used. The MOPSO without personal best is another promising idea, mostly because of its simplicity, but additional experiments are needed to investigate if these good results are applicable to other MOPSOs or if they scale to MaOPs.

Other interesting ideas are presented by MOPSO/D, mostly the use of dominance and decomposition simultaneously, however the simplicity of the MOPSO framework is somewhat lost in this algorithm, since several modifications are implemented. R2-MOPSO on the other hand, presents a simple, indicator-based MOPSO. It uses the scalable R2 indicator, however the dominance relation is not used to guide the search. Associating dominance with the R2 indicator values would be an interesting idea, especially for few objectives.

2.2 MOPSOs employing multiple swarms

A MOPSO based on region decomposition is proposed by Chen and Liu (2014) and called MOPSO-M2M. This algorithm decomposes the objective space into a number of subregions and searches these subregions using sub-swarms. Each subregion has a set of weight vectors, and each weight vector is associated with a particle, this weight vector is used with the Tchebycheff scalarizing function to select the personal best leader for each particle. The global best solution of each particle is selected at random from the repository of its sub-region. The only communication between the sub-swarms is through the offspring solutions, which are combined, and then split among the sub-regions following the M2M (Liu et al., 2014) scheme. Experimental results indicate that MOPSO-M2M outperforms two MOPSOs from the literature.

A MOPSO based on multiple swarms and reference points is proposed by Britto and Pozo (2015) and called R-Multi. This algorithm is inspired by the previously proposed I-Multi (Britto et al., 2013) and maintains a set of swarms, where each one uses an archiver defined by a reference

point. In this archiver, once the repository becomes full, the solution farthest from the reference point is removed. The swarms communicate in a ring topology, where a sub-swarm tries to insert all the non-dominated solutions from its neighbors in its own repository. This exchange of information occurs in a predefined interval, defined in number of iterations. The results indicate that R-Multi is competitive in terms of diversity, especially when a high number of objectives is considered.

A MOPSO based on co-operative multiple swarms is proposed by Maltese et al. (2015) and called CVEPSO. This algorithm is a combination of VEPSO (Parsopoulos and Vrahatis, 2002) and CPSO-S (van den Bergh et al., 2000) along with two variants of it. VEPSO solves a MOP by assigning a single-objective PSO to each objective of the MOP, the communication between the sub-swarms occurs by a particle selecting its global best as the best solution from a different sub-swarm. CPSO-S is a single-objective multi-swarm PSO that assigns a sub-swarm to each decision variable of the problem. CVEPSO combines VEPSO and CPSO-S, by simply replacing the standard PSO used by each swarm of VEPSO by a CPSO-S variant. The results indicate that CVEPSO outperforms VEPSO. Moreover, when compared to other MOPSOs, CVEPSO scaled well in terms of hypervolume, but poorly in terms of solution distribution. Overall, CVEPSO is competitive with other MOPSOs, performing equally or better in most problems.

Another study involving VEPSO is presented by Scheepers and Engelbrecht (2016). In this work, the authors identify that VEPSO continues to explore and does not focus enough on exploitation. To deal with this issue, it is proposed a new MOPSO called MGVEPSO$_A$. In this algorithm, an archive guide term is added to the velocity equation of PSO. This new guide is randomly selected from the repository of non-dominated solutions. The results indicate that in general, MGVEPSO$_A$ outperformed VEPSO due to increased exploitation.

Regarding these multi-swarm MOPSOs, MOPSO-M2M and R-Multi presents a very important similarity: they employ reference point based MOPSOs, where each sub-swarm is responsible for finding solutions around a reference point. This approach can be superior to the clustering strategy employed by I-Multi, because if the solutions of the initial front are not diverse enough, the swarms are not well spread along the search space and the performance of the algorithm can be severely deteriorated. In Chapter 7 we implement a similar idea through the use of decomposition.

CVEPSO and MGVEPSO$_A$ share a common characteristic as well, they are both based on the VEPSO framework. The VEPSO can be thought as a reference point based multi swarm, as the two previously presented MOPSOs, but using only extreme reference points. This fact discourage us from employing the VEPSO framework, however the ideas of using co-operative sub-swarms or a global best chosen from the repository of a different sub-swarm can be adapted to the frameworks of MOPSO-M2M or R-Multi.

2.3 MOEAs for MaOPs

An evolutionary algorithm based on a knee point is proposed by Zhang et al. (2015) and called KnEA. Its framework is based on NSGA-II (Deb et al., 2000), however during the mating selection, three criteria are used: the dominance relationship, a knee point criterion and a weighted distance. A non-dominated solution is preferred over a dominated one, a knee point is preferred over a non knee point, as last criterion, a weighted distance is calculated. This distance is a weighted sum of the k nearest neighbors of a solution. The weight is calculated based on the distance of the neighbor to the center of all considered neighbors. After the non-dominated sorting, an adaptive strategy is conducted to identify the solutions located in the knee regions of

each non-dominated front in the combined population. Finally an environmental selection is performed to select the parent population of the next generation. This selection uses dominance as primary criterion and as second criterion, the distance from the solutions to an extreme hyperplane defined by the solutions having the maximum value in each objective. The results indicate that KnEA achieves a competitive performance compared to state-of-the-art MOEAs.

A decomposition based MOEA is proposed by Asafuddoula et al. (2015) and called I-DBEA. A new method for calculating the scalarized fitness of the solutions is proposed, based on two measures, the first (d_1) measures convergence and the second (d_2) indicates diversity, similar to the PBI (Zhang and Li, 2007). Moreover, a normalization procedure similar to that of NSGA-III is proposed. A new replacement methods is proposed as well, where if a solution is non-dominated regarding the population, it tries to replace any individual in the population by comparing their d_2 distances, and in case of tie, by comparing their d_1 distances. The results indicate that I-DBEA is able to deal with unconstrained and constrained many-objective problems better than or at par with existing state-of-the-art algorithms.

An evolutionary algorithm based on the well-known NSGA-II is proposed by Deb and Jain (2014) and called NSGA-III. The main difference between NSGA-II and NSGA-III remains in the selection mechanism, as the latter uses a set of well-spread reference points and emphasizes population members associated with each reference point, but penalizes members associated with crowded reference points. An online normalization mechanism is proposed to deal with test problems having differently scaled objective values. NSGA-III was compared to different MOEA/D (Zhang and Li, 2007) variants in different scenarios, and in general, achieved competitive or better results.

A MOEA/D (Zhang and Li, 2007) variant based on dominance and decomposition simultaneously is proposed by Li et al. (2015) and called MOEA/DD. The main difference between MOEA/D and MOEA/DD is in the update procedure, as in MOEA/D the best solution for a subproblem can only be replaced by a new solution if it has a better scalarizing value than the former. The update procedure introduced by MOEA/DD combines dominance and decomposition, where it keeps even dominated solutions if they belong to an isolated sub-region, however, in crowded sub-regions, non-dominated solutions are preferred. To distinguish between non-dominated solutions, density estimation and scalarizing values are used. Experimental results indicate that MOEA/DD outperforms state-of-the-art optimizers in most of the investigated problems.

Among the MOEAs for MaOPs included here, KnEA is the only one that does not employ decomposition or reference points. Unlike the other algorithms it is guided to the knee point of the Pareto front. This strategy allows the algorithm to achieve good convergence to an often preferred region of the Pareto front, however, the diversity of solutions is compromised, hence the algorithm applicability is reduced.

The other three MOEAs reviewed employ decomposition or reference points to guide the search, which is a trending approach to deal with MaOPs. The last two approaches employ dominance and decomposition simultaneously, which can improve the search ability of such algorithms, especially for dealing with few objectives, where the Pareto dominance is more important. The update strategy proposed for MOEA/DD was employed in Chapter 7 of this thesis, mostly because it is simpler than NSGA-III, and only requires small changes in the MOEA/D framework.

2.4 Multi-objective optimizers based on hyper-heuristics

A hyper-heuristic based on choice function for multi-objective problems is proposed by Maashi et al. (2014) called HH_CF. This algorithm selects a low-level heuristic using a choice function that takes under consideration a combination of four metrics: Algorithm effort (AE), Ratio of non-dominated individuals (RNI), hypervolume and Uniform distribution of a non-dominated population (UD). Three evolutionary algorithms are used in this work as low-level heuristics: NSGA-II (Deb et al., 2000), SPEA2 (Zitzler et al., 2002) and MOGA (Fonseca and Fleming, 1998). The move acceptance used is all moves. The results from the experimental studies indicate that HH_CF is able to outperform the low-level heuristics used alone, moreover, they outperform a random hyper-heuristic and the AMALGAM (Vrugt and Robinson, 2007) method on two objective problems, however in a real-world three objective problem, AMALGAM outperformed HH_CF in terms of hypervolume.

An AOS based on MOEA/D is proposed by Li et al. (2014) and called FRRMAB. The proposed method consists of two modules: one for credit assignment and the other is for operator selection. The credit assignment attributes a reward for a recently used operator based on the improvement caused by its application on each subproblem, stored in a sliding window with a decay mechanism. The operator selection is based in the UCB (Auer et al., 2002) algorithm created to solve the multi-armed bandit problem. The AOS was used to select between four DE mutation operators. The results indicate that FRRMAB is robust and can significantly improve the performance of MOEA/D.

A MOEA/D algorithm using a novel choice function hyper-heuristic is proposed by Gonçalves et al. (2015b) and called MOEA/D-HH. This algorithm proposes an adaptive choice function to determine the low-level heuristic that should be applied to each subproblem. The reward in this function is calculated based on the scalarized fitness difference between a parent and its child solution. The low-level heuristics are DE operators. The experimental results indicate that MOEA/D-HH improves the performance of the MOEA/D using single low-level heuristics. Moreover, MOEA/D-HH was favorably compared to state-of-the-art methods.

A new optimizer based on hyper-heuristic is proposed by Guo et al. (2016) and called HMOEA. In this algorithm, the total population is divided in several subgroups and a different multi-objective algorithm is assigned to optimize each group. The population size assigned to each group is adjusted at each generation according to the performance of each algorithm, hence the best performing optimizers have a larger population. The poorly performing algorithms are assigned a small population instead of zero, so they are not removed from the search. The performance of the optimizers is evaluated using the coverage rate (Zitzler and Thiele, 1999) and the population size of a group is defined based on the relative quality of the Pareto front obtained by its optimizer regarding all Pareto fronts. The employed optimizers are NSGA-II (Deb et al., 2000), MOPSO (Suresh et al., 2007) and MOBBO (Guo et al., 2016). In most of the problems, HMOEA loses for MOPSO, however, it outperforms the other optimizers.

The heuristic selection methods of HH_CF and FRRMAB were already employed in this thesis, in Chapter 4. Another interesting alternative to employ the heuristic selection of HH_CF is in the MOEA/D-HH framework, which is a promising improvement for decomposition based methods. The idea proposed in HMOEA is a good alternative to merge algorithms, especially if the algorithms to be used as low-level heuristics are very different from each other and are able to achieve good results in different difficult problem domains.

2.5 EDAs for MOPs

In this section we discuss state-of-the-art EDAs from the literature for continuous MOPs and MaOPs. EDAs for combinatorial problems are not discussed in this section, but the interested reader is referred to (Zangari et al., 2017a,b) for recent approaches.

A MOEA/D variant employing a CMA-ES (Hansen and Ostermeier, 1996) model for each subproblem is proposed by Zapotecas-Martínez et al. (2015) and called MOEA/D-CMA. In an iteration of this algorithm, for each subproblem, a set of solutions is sampled, so these solutions can be used to update the best solutions of its neighbors as in MOEA/D. Then, the CMA-ES parameters of each subproblem are updated using the best solutions generated by its own distribution and solutions injected from its neighbors. Experimental results indicate that MOEA/D-CMA presents competitive results when compared to MOEA/D (Zhang and Li, 2007).

An unbounded population multi objective CMA-ES (Hansen and Ostermeier, 1996) is proposed by Krause et al. (2016) and named UP-MO-CMA-ES. In this algorithm, all the non-dominated solutions are added to the population, whose members have a CMA-ES instance. To select the parent individuals, an extreme solution is picked with a fixed probability, if no extreme point is picked or it seems to have converged, because its step size is too small, an interior point is picked with a probability based on its hypervolume contribution. Then the covariance matrix of this individual is combined with the covariance matrices of its two immediate neighbors. Following a new point is sampled from the updated covariance matrix. An experimental study was conducted using the COCO framework (Brockhoff et al., 2016), and the results indicate that the performance of UP-MO-CMA-ES is promising, unless it faces highly multi-modal problems.

A MOEDA based on neural network is proposed by Martí et al. (2016) and called MONEDA. This algorithm combines the NSGA-II fitness assignment and a model builder proposed by the authors, called MB-GNG. This model builder is a modification of the growing neural gas network (GNG). MB-GNG is a one-layer network that defines each class as a local Gaussian density and adapts them using a local learning rule. The network can be interpreted as a Gaussian mixture. The experimental results show that MONEDA performs similarly to other approaches in problems with relatively few dimensions (in objective space), however as the dimensionality scales up, MONEDA outperforms other MOEAs and MOEDAs in terms of quality of solutions and computational complexity.

A MOEDA based on clustering and Gaussian models is proposed by Lin et al. (2016) and called AMEDA. In the model building of AMEDA, the population is partitioned into a number of local clusters, then a global cluster is built by randomly selecting a solution from each local cluster. Following, the covariance matrices of the local clusters and the global cluster are calculated. Next, each solution is attributed its local cluster covariance matrix, or the global with a probability that is defined using an adaptive mechanism. The experimental results indicate that AMEDA dramatically outperforms representative MOEAs from the literature on the tested problems.

In this thesis, we already extended the idea proposed in MOEA/D-CMA by employing the update mechanism of MOEA/DD. The ideas proposed for UP-MO-CMA-ES of using an unbounded population and updating the models using information from the neighbors were also employed in Chapter 7. The model building mechanisms proposed in the last two works are promising, however they are very recent and could not be considered for this work, future works include investigating these mechanisms.

2.6 Discussion

This chapter presented recently proposed works in the areas of MOPSOs, multi-swarm MOPSOs, many-objective MOEAs, hyper-heuristic approaches for MOPs and MOEDAs. In an overall observation, we can highlight the amount of different, promising proposals for tackling MOPs and MaOPs. This variety of research lines opens a great amount of possibilities of combining and/or modifying the proposed ideas to create promising approaches. Moreover, this variety or works indicates that there is much interest in the research community in seeking alternatives to deal with MOPs and MaOPs.

Chapter 3

Background

This chapter presents the theoretical foundation used throughout this work. Firstly some concepts of optimization problems are presented in Section 3.1. The Particle Swarm Optimization (PSO) algorithm and the most commonly used mechanisms to make it able to deal with multi-objective problems are presented in Sections 3.2 and 3.3 respectively. Another well-known framework for solving multi-objective problems, the multi-objective evolutionary algorithm based on decomposition (MOEA/D) is presented in Section 3.4, and a state-of-the-art algorithm based on the MOEA/D framework that simultaneously incorporates dominance and decomposition (MOEA/DD) is presented in Section 3.5. Other promising approaches also used in this work are Estimation of Distribution Algorithms (EDAs) and hyper-heuristics, which are presented respectively in Sections 3.6 and 3.7. The benchmark problems we use in the experimental studies to test the algorithms are presented in Section 3.8, but to evaluate and compare different algorithms on these multi-objective problems, we need performance metrics, the ones used in this work are presented in Section 3.9. To make sure that the differences in results obtained with different algorithms are not random, we use the statistical tests described in Section 3.10. Finally a brief discussion of the concepts presented in this chapter is available in Section 3.11.

3.1 Optimization Problems

Optimization problems arise naturally in most disciplines, and their solution has challenged researchers for some time. This class of problems can be characterized by single-objective and multi-objective problems. Single objective problems are being successfully solved by bio-inspired algorithms like Particle Swarm Optimization (PSO) (Kennedy and Eberhart, 1995) and Genetic Algorithms (GA) (Holland, 1975) over the last decades. Multi-objective problems also have been successfully solved by classical multi-objective versions of these optimizers, like Speed-constrained Multi-objective PSO (SMPSO) (Nebro et al., 2009) and Non-dominated Sorting Genetic Algorithm II (NSGA-II) (Deb et al., 2000).

The next sections present definitions of such problems, where Section 3.1.1 defines single-objective optimization problems, while Section 3.1.2 defines the multi-objective ones.

3.1.1 Single-objective problems

Single-objective optimization problems are addressed by many search techniques in the literature (Holland, 1975; Kennedy and Eberhart, 1995). A general unconstrained single-objective optimization problem can be defined as optimizing (maximizing or minimizing) $f(\boldsymbol{x})$, where $\boldsymbol{x} = (x_1, ..., x_n)$ is an n-dimensional variable vector from some universe Ω (Coello et al., 2006).

Ω contains all possible $\boldsymbol{x}$ that can be used to satisfy an evaluation of $f(\boldsymbol{x})$. $\boldsymbol{x}$ can be a vector of continuous or discrete variables. The method for finding the global optimum (possibly more than one) of any function is referred to as global optimization method (Coello et al., 2006).

Figure 3.1 visually shows the concept of single-objective optimization, where the solutions $\boldsymbol{x}$ in the feasible decision space Ω (represented by black points) are translated into the objective space $f(\boldsymbol{x})$. The optimal solution (minimization) is represented by a black point and the sub-optimal ones by white points, this ranking is made according to the objective value.

3.1.2 Multi-objective problems

An unconstrained MOP can be defined as optimizing $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), ..., f_m(\boldsymbol{x}))$, these objectives may be linear or non-linear and continuous or discrete in nature. The evaluation

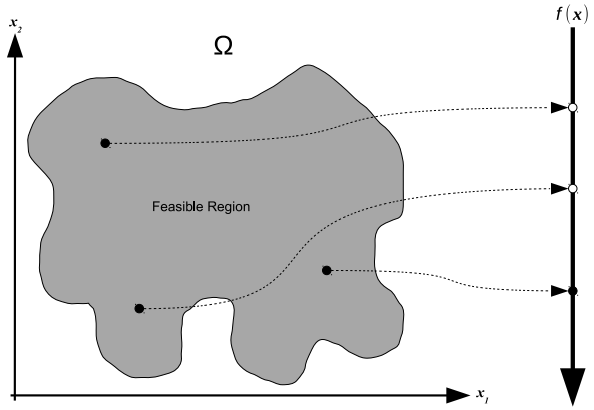


Figure 3.1: Representation of the decision and objective spaces

function, $\boldsymbol{f} : \Omega \rightarrow \Lambda$ is a mapping from the vector of decision variables $\boldsymbol{x}$ to output vectors $\boldsymbol{u}$, where $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{u} = (u_1, ..., u_m)$ (Coello et al., 2006).

Due to the multiple objective functions, the notion of "optimum" changes, because in MOPs the aim is to find good compromises (trade-offs) between the values of the objective functions instead of a single solution. The most usual notion of optimum adopted is the Pareto optimum where a solution $\boldsymbol{x} \in \Omega$ is said to be Pareto optimal with respect to Ω iff there is no $\boldsymbol{x}' \in \Omega$ for which $\boldsymbol{u}' = \boldsymbol{f}(\boldsymbol{x}')$ dominates $\boldsymbol{u} = \boldsymbol{f}(\boldsymbol{x})$ (Coello et al., 2006).

To determine if a vector dominates another, the definition of Pareto dominance is employed, where a vector $\boldsymbol{u}$ is said to dominate another vector $\boldsymbol{u}'$, denoted by $\boldsymbol{u} \prec \boldsymbol{u}'$, in case of minimization, iff $\boldsymbol{u}$ is partially less than $\boldsymbol{u}'$, i.e., $\forall i \in \{1, ..., m\}, u_i \leq u'_i \wedge \exists i \in \{1, ..., m\} : u_i < u'_i$ (Coello et al., 2006).

A Pareto optimal solution is taken to mean with respect to the entire decision variable space unless otherwise specified. The collection of all Pareto optimal solutions regarding a decision space is known as Pareto optimal set, denoted by $\mathcal{P}^*$. The image of $\mathcal{P}^*$ in the objective space is termed Pareto front, denoted by $\mathcal{PF}^*$ (Coello et al., 2006).

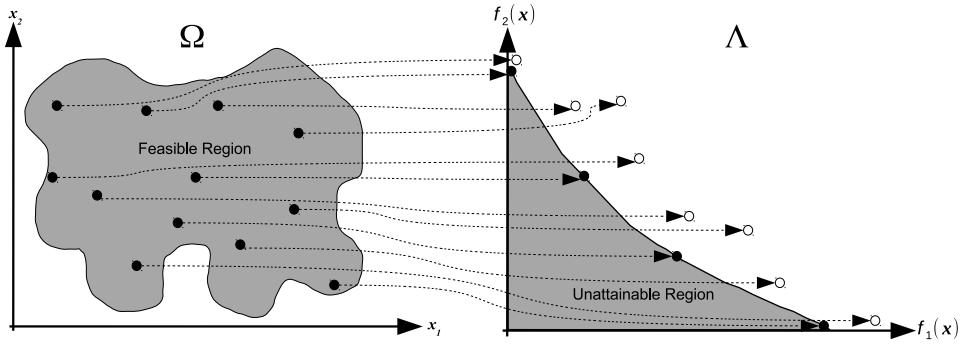


Figure 3.2: Representation of Pareto front, dominated and non-dominated solutions

In some optimization problems, especially benchmarks, the true Pareto front is known mathematically (i.e., the best compromise solutions that can be achieved in that problem). This true Pareto front is represented by PF_t throughout this work. Since the multi-objective optimizers are only able to provide an approximation of the true Pareto front, to differentiate, such approximated front (known by the optimizer) will be denoted as PF_k in this work.

Figure 3.2 summarizes the concepts presented in this section. In its left side, there is the feasible decision space represented by the gray area, with several black points representing

feasible solutions. The dotted arrows are the mapping $\Omega \rightarrow \Lambda$ from the decision space to the objective space. In the right side of the figure, there is the objective space; the gray area represents the unattainable region of the objective space, while the white area is the attainable and the black line separating both is the true Pareto front (PF_t). The black points on this side represent the non-dominated vectors, collectively forming the known Pareto front (PF_k), while the white points are the dominated solutions.

MOPs with two or three objectives are being successfully solved by state-of-the-art optimization algorithms since the beginning of the nineties. However, in most real-world problems involving multiple stakeholders and functionalities, there often are more than three objectives involved (Deb and Jain, 2014).

Problems involving more than three objective functions to be optimized simultaneously are known as Many-Objective Optimization Problems (MaOPs) and several studies (Britto and Pozo, 2012; Deb and Jain, 2014; Ishibuchi et al., 2008; Nebro et al., 2009) indicate that the search ability of Pareto-based algorithms is severely deteriorated in such type of problems.

The area that studies solutions to deal with MaOPs is known as Many-Objective Optimization and had been one of the main research areas in the multi-objective optimization community for the past few years (Deb and Jain, 2014; Ishibuchi et al., 2008).

The main difficulties identified by the increase in the number of objectives are (Ishibuchi et al., 2008):

1. Deterioration of the search ability: When the number of objective increases, almost all solutions become non-dominated. This severely weakens the selection pressure toward the Pareto front. That is, the convergence property of the algorithms is severely deteriorated.

2. Exponential increase in the number of solutions required for approximating the entire Pareto front. The goal of a multi-objective optimizer is to find a set of non-dominated solutions that well approximates the entire Pareto front. Since the Pareto front is a hyper-surface in the objective space, the number of solutions required for its approximation exponentially increases with the dimensionality of the objective space.

3. Difficulty for visualization of the solutions. It is usually assumed that the choice of a final solution from a set of obtained non-dominated solutions is done by a decision maker based on his/her preference. The increase in the number of objectives makes the visualization of obtained non-dominated solutions very difficult. This means that the choice of a final solution becomes very difficult in many-objective optimization.

Various approaches have been proposed to the handling of MaOPs in the literature, and those can be categorized as follows (Ishibuchi et al., 2011):

1. Dimensionality reduction: try to decrease the number of objectives by removing unnecessary objectives. If the number of objectives in a MaOP can be reduced to two or three, traditional algorithms may work well on them (Saxena et al., 2013; von Lücken et al., 2015).

2. Preference incorporation: use a decision maker preference to realize efficient multi-objective search by concentrating on preferred regions of the Pareto front (Landa et al., 2013; López-Jaimes and Coello, 2014).

3. Selection pressure enhancement: includes various proposals for increasing the selection pressure toward the Pareto front (Tomita et al., 2015; Zhu et al., 2016).

4. Different fitness evaluation schemes: These approaches do not use Pareto dominance for fitness evaluation. Quality indicators and scalarizing functions have been used for fitness evaluation (Bader and Zitzler, 2011; Beume et al., 2007; Zhang and Li, 2007).

In this work we aim to create multi-objective optimizers able to deal with both: multi and many-objective problems. To accomplish this, we use both: approaches already proposed in the literature, but applied in other algorithms as well as new approaches specially developed to improve the results of the optimizers.

3.2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a stochastic optimization technique developed by Kennedy and Eberhart (1995). PSO belongs to the class of swarm intelligence algorithms, however it is similar to evolutionary algorithms in some points, like the population-based nature and the initialization of its population with random solutions. However PSO is different in other points, like being modeled based in the swarming and flocking behaviors in animals instead of Darwinian evolution. Traditionally, unlike other population-based methods, PSO does not resample populations to produce new ones: it has no selection of any kind. Instead, PSO maintains a single static population whose members are tweaked in response to new discoveries about the space (Eberhart and Shi, 2001; Luke, 2013).

In PSO simulation, the behavior of each individual (or particle) is affected by either the best local, or the best global individual. The approach uses then the concept of population (or swarm) and a measure of performance similar to the fitness value used with evolutionary algorithms. Also, adjustments are made to the solutions, however using the PSO equations instead of a crossover operator. Note that PSO allows individuals to benefit from their past experiences whereas in an evolutionary algorithm, normally the current population is the only "memory" used by the individuals. PSO has been successfully used for both continuous nonlinear and discrete optimization (Bansal and Deep, 2012; Coello et al., 2006; de Carvalho and Pozo, 2014).

PSO operates almost exclusively in multidimensional metric and usually real-valued spaces. This is because its candidate solutions are mutated toward the best discovered solutions so far, which really needs a metric space (Luke, 2013).

Because of its use in real-valued spaces, and because PSO is inspired by flocks and swarms, PSO practitioners tend to refer to candidate solutions not as population of individuals but as a swarm of particles. These particles never die (there is no selection), instead, they move in the search space. A particle consists of two parts (Luke, 2013):

- The particle location in space, $\boldsymbol{x} = (x_1, ..., x_n)$. This is equivalent of the individual genotype in evolutionary algorithms.
- The particle velocity $\boldsymbol{v} = (v_1, ..., v_n)$. This is the velocity and direction at which the particle is traveling at each iteration.

To update the position of the particle $\boldsymbol{x}_i$ at the iteration t, PSO employ the Equation (3.1) (Durillo et al., 2009):

$$\boldsymbol{x}_i^{(t)} = \boldsymbol{x}_i^{(t-1)} + \boldsymbol{v}_i^{(t)} \tag{3.1}$$

where the current velocity $\boldsymbol{v}_i^{(t)}$ is given by Equation (3.2):

$$\boldsymbol{v}_i^{(t)} = \underbrace{\omega \times \boldsymbol{v}_i^{(t-1)}}_{\text{inertial}} + \underbrace{C_1 \times r_1 \times (\boldsymbol{x}_{b_i} - \boldsymbol{x}_i^{(t-1)})}_{\text{cognitive}} + \underbrace{C_2 \times r_2 \times (\boldsymbol{x}_{g_i} - \boldsymbol{x}_i^{(t-1)})}_{\text{social}} \tag{3.2}$$

In Equation (3.2), $\boldsymbol{x}_{b_i}$ is the best solution found so far by $\boldsymbol{x_i}(t)$, $\boldsymbol{x}_{g_i}$ is the best particle (also known as global leader) that the neighborhood of $\boldsymbol{x}_i$ has found, ω is the inertia weight of the particle and controls the trade-off between using global and local information, r_1 and r_2 are two uniformly distributed random numbers in the range [0,1], and C_1 and C_2 are specific parameters which control the influence of personal and global best particles.

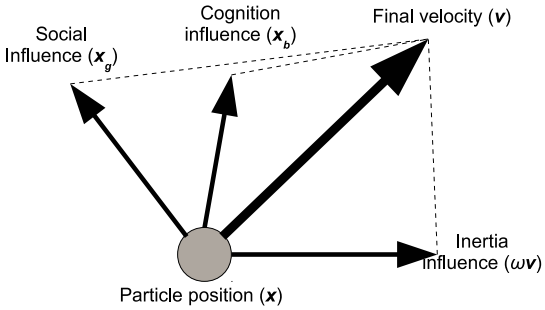


Figure 3.3: Representation of the velocity equation

Shi and Eberhart (1998) have pointed that the right side of Equation (3.2) consists of three parts: the first is the inertial part, which represents the previous velocity of the particle, the second is the cognitive part, which represents the private thinking of the particle itself, and the third is the social part which represents the collaboration among the particles. This velocity update scheme is represented in Figure 3.3.

Algorithm 1: Pseudocode of a PSO algorithm

```
1  begin
2      Initialize swarm
3      Locate leader
4      t=0
5      while t < t_max do
6          for each particle do
7              Update Position // flight (Equations (3.1) and (3.2))
8              Evaluation
9              Update Personal Best
10         end
11         Update Global Leader
12         t++
13     end
14 end
```

Algorithm 1 describes the pseudocode of a general single-objective PSO. First, the swarm is initialized. This initialization includes positions and velocities. The corresponding $\boldsymbol{x}_b$

of each particle is initialized and the global leader is located (x_g). Then, for a maximum number of iterations (t_{max}), each particle flies through the search space updating its position, evaluating this position, and updating its personal best. At the end of each iteration, the global leader is updated (Durillo et al., 2009; Reyes-Sierra and Coello, 2006).

3.3 Multi-objective PSO

PSO has several characteristics that make it suitable for multi-objective optimization, like being population-based, easy to understand and achieving fast convergence. However, to extend a PSO into a MOPSO some modifications need to be done to cope with the fact that the solution of a multi-objective problem is not a single one, but a set of non-dominated solutions. These modifications involve considering at least two issues:

1. How to retain the non-dominated solutions found during the search? Also it is desirable that these solutions are well spread along the front.

2. How to select particles to be used as leaders?

To deal with the first issue, usually an external archive or repository is employed, as is done in some multi-objective evolutionary algorithms (MOEAs). This repository is used to store the best non-dominated solutions found so far, also, its content is usually reported as the final output of the algorithm. Ideally all non-dominated solutions would be included in the repository, however by doing this, its size can increase quickly. To avoid this problem, usually the size of the repository is limited by an upper bound. In this scenario, when the repository is full and a new non-dominated solution is found, a criterion is needed to decide if the new solution is included, and in such case, which existing solution has to be removed. Several criteria were proposed in the literature and called archivers. Some important archivers are described in Section 3.3.1.

Dealing with the second issue is not a trivial task, since each particle needs to select its own global leader, usually from the repository that contains a set of equally good candidates. Moreover, the selection of leaders can have a strong impact on the performance of a MOPSO, since it can influence the convergence and diversity characteristics of the algorithm. Several leader selection methods were proposed in the literature to deal with this issue, and some important methods are presented in Section 3.3.2.

Algorithm 2 shows the pseudocode of a MOPSO. First, the swarm is initialized (position and velocity of the particles). Then, the typical approach is to use a repository to store the leaders, which are taken from the non-dominated particles in the swarm. At each iteration, for each particle, a leader is selected and the flight is performed. Most of the existing MOPSOs apply some sort of turbulence (or mutation) operator after performing the flight. Then the particle is evaluated and its corresponding personal best is updated. A new particle replaces its personal best particle usually when this particle is dominated or if both are incomparable (non-dominated among themselves). After all the particles have been updated, the repository is updated as well. After the termination condition, the archive is returned as the result of the search (Durillo et al., 2009; Reyes-Sierra and Coello, 2006).

3.3.1 Archiving methods

According to Section 3.3, usually MOPSOs use an external archive to store the non-dominated solutions found during the search, however the number of such solutions can increase

Algorithm 2: Pseudocode of a MOPSO algorithm

```
1  begin
2      Initialize swarm
3      Initialize repository
4      t=0
5      while t < t_max do
6          for each particle do
7              Select leader
8              Update Position // flight (Equations (3.1) and (3.2))
9              Turbulence
10             Evaluation
11             Update Personal Best
12         end
13         Update external archive
14         t++
15     end
16     Report results in the external archive
17 end
```

quickly, especially when dealing with many-objectives. This increase in the size of the repository makes its maintenance computationally expensive, also the memory usage to store these solutions can increase largely.

To avoid such problems usually the size of the archive is bounded, so it only keeps the N "best" solutions. However the question is what happens when the archive is full and a new non-dominated solution is generated. The archiver must decide whether or not to include the new solution in the archive, and if it does include, which existing solution(s) should be removed. In this way, although ideally we would want that the archive maintains the "best" solutions found so far, it can only, in general, maintain an approximation to that (Corne and Knowles, 2003).

A formalization of a simple archiver, called precise archiver is presented by Corne and Knowles (2003). A precise archiver is one that ensures that as many non-dominated points as possible are kept in the archive, and only non-dominated points are kept. A precise archiver with size limited to N solutions can be represented by Algorithm 3 (Corne and Knowles, 2003; López-Ibáñez et al., 2011).

In this algorithm, firstly it is verified if the new solution to be inserted ($\boldsymbol{u}$) dominates any of the solutions previously stored, if it is the case, the dominated solutions are removed and $\boldsymbol{u}$ is inserted. The function $nonDom()$ returns only the non-dominated solutions from the input. If $\boldsymbol{u}$ does not dominate any of the solutions stored in $PF_k^{(t-1)}$ and the size of $(PF_k^{(t-1)} \cup \boldsymbol{u})$ is inferior to the limit, $\boldsymbol{u}$ is included. If $\boldsymbol{u}$ is dominated by any member of $PF_k^{(t-1)}$, the repository remains unchanged. Finally, if $\boldsymbol{u}$ is non-dominated, does not dominate any solution in $PF_k^{(t-1)}$ and the repository is full, an archiving method $filter()$ is used to determine if $\boldsymbol{u}$ enters to the repository, and if it does, which non-dominated solution must be removed from it.

Britto and Pozo (2012) have compared several archiving methods: some of them do not use the precise archiver scheme, others do not bound the size of the repository. Three of these archivers that present good results, use the precise archiver, and limit the size of the repository, are reviewed in this section. The Crowding Distance (CD) method is presented in Section 3.3.1.1,

Algorithm 3: Precise archiver

Input: $PF_k^{(t-1)}, \boldsymbol{u}, N$ (Previous archive, new solution, maximum archive size)
Output: $PF_k^{(t)}$
begin
  if $\exists \boldsymbol{u}' \in PF_k^{(t-1)}, \boldsymbol{u} \prec \boldsymbol{u}'$ **then**
    $PF_k^{(t)}$=$nonDom(PF_k^{(t-1)} \cup \boldsymbol{u})$
  else
    if $|nonDom(PF_k^{(t-1)} \cup \boldsymbol{u})| < N$ **then**
      $PF_k^{(t)} = nonDom(PF_k^{(t-1)} \cup \boldsymbol{u})$
    else
      $PF_k^{(t)} = filter(PF_k^{(t-1)} \cup \boldsymbol{u})$
    end
  end
end

Section 3.3.1.2 presents the Multilevel Grid Archiving (MGA) and the Ideal archiver is presented in Section 3.3.1.3.

3.3.1.1 Crowding Distance

The Crowding Distance (CD), proposed by Deb et al. (2000), is a diversity estimator that has been extensively applied to evolutionary multi-objective algorithms to promote diversity (Padhye et al., 2009). This metric is used to estimate the density of solutions surrounding a particular point. To calculate the CD, we sort the solutions regarding each objective, while the solutions are sorted, the average distance of its two neighbors (regarding the current objective) is calculated. At the end, this averaged value is used as an estimation of the size of the largest cuboid enclosing the solution $\boldsymbol{u}$ without including any other solution of PF_k.

Algorithm 4: Crowding Distance calculation

$N = |PF_k|$
for *i=1 to N* **do**
  $CD_i = 0$
end
for $j = 1$ *to* m **do**
  PF_k =sort(PF_k, j)
  $CD_1 = CD_N = \infty$
  for $i = 2$ *to* $(N-1)$ **do**
    $CD_i = CD_i + (u_{j,(i+1)} - u_{j,i-1})$
  end
end

The CD for each solution $\boldsymbol{u}$ in the set PF_k is calculated through Algorithm 4, where m is the number of objectives, N is the number of solutions in PF_k, and $sort(a, b)$ is used to sort the elements of a in the ascending order in relation to its b-th objective value. $u_{j,i}$ represents the j-th objective value of the i-th solution of the set PF_k.

The algorithm starts with N storing the number of solutions contained in PF_k, then the CD of each solution is initialized to 0. The next step is, for each objective, the population is sorted in an ascending order in relation to this objective. The first and last solutions (extreme solution of the axis) have its CD value set to ∞. Next, CD of the other solutions is calculated, where at each objective, the CD_i is increased by the difference between the objective values of the neighbors of u_i.

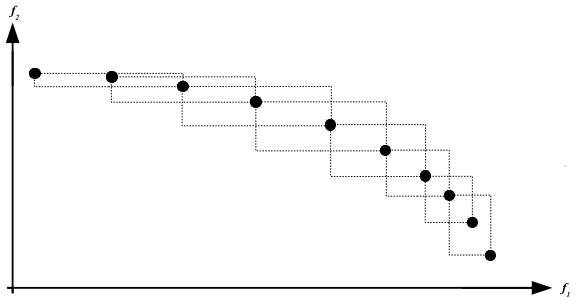


Figure 3.4: Representation of the CD

Figure 3.4 represents the CD of the non-dominated solutions in a PF_k, where each of such solutions (represented as black points) present a cuboid drawn around them in dotted lines. Note that the extreme solutions do not have a cuboid, this is because their CD is considered to be infinity.

In the archiving procedure, first the new solution is inserted in the repository (its size becomes $N + 1$), then the solution in the repository with the smallest CD value is removed, since this solution is located in a more crowded region. This is the archiver originally used in SMPSO (Nebro et al., 2009).

3.3.1.2 MGA

The Multi-level Grid Archiving (MGA), proposed by Laumanns and Zenklusen (2011), works by dividing the objective space into boxes, in such a way that every solution in the repository has a box index. Then, the dominance relation between the boxes is observed. If the new solution to be added belongs to one of the dominated boxes, it is not included, otherwise one of the solutions inside a dominated box is randomly removed. If there is no dominance relation between the boxes, the objective space is split again into smaller boxes until at least one dominated box is found.

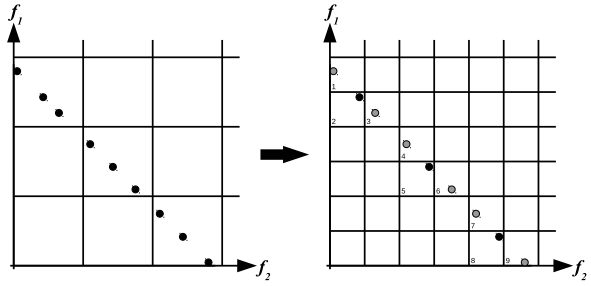


Figure 3.5: Representation of the MGA

Figure 3.5 represents the MGA method for a bi-objective minimization problem. In the left grid, the objective space is divided into larger boxes, and there is no box containing a

solution that is dominated by another box. Hence the objective space is split again in smaller boxes, presented in the right grid where boxes 2, 5 and 8 dominate the boxes 1, 3, 4, 6, 7 and 9. In this case, MGA archiver will randomly remove a solution from any of the dominated boxes.

3.3.1.3 Ideal

The Ideal archiver, proposed by Britto and Pozo (2012), guides the solutions in the archive to a specific area of the objective space to increase the convergence of the search. To do so, the Ideal point is selected as guide. The ideal point is obtained with the best value for every objective function, obtained at each iteration between the points in the external archive. In this approach, the distance to the ideal point is used to define which solutions will remain in the archive.

In this archiver, firstly the approximated ideal point is obtained from the repository, then the Euclidean distance from each point in the archive to the ideal point is calculated. After that, the point with the highest distance is removed. The hypothesis behind this method is that guiding the selection of the points in the archive to a region close to the ideal point will increase the convergence of the search to the Pareto front and will place the solutions in a good area of the objective space.

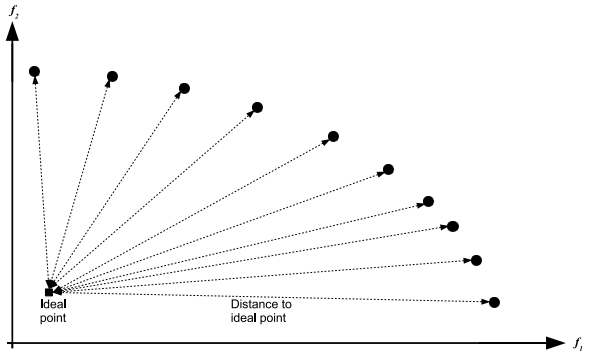


Figure 3.6: Representation of the Ideal archiver

Figure 3.6 shows the general idea of the Ideal archiver, where the square point represents the Ideal vector and the black points represent the solutions in the repository. The dotted arrows represent the Euclidean distance between the solutions and the ideal vector. The solution with the highest distance from the ideal vector is removed from the repository.

3.3.2 Leader selection methods

Selecting leaders in a MOPSO is not trivial, as commented in Section 3.3, the leader of each particle has to be selected from a set of equally good solutions. The selection of the leader is an important task, since it is expected that the leaders guide the search toward better regions, also they need to provide diversity for the search, avoiding the convergence to a single point. In the literature there are several methods for leader selection and a comparison among some of them was performed by Castro Jr. et al. (2012).

In this section four leader selection methods that presented good results in (Castro Jr. et al., 2012) are presented: Crowding Distance (CD) in Section 3.3.2.1, Weighted Sum (WSum) in Section 3.3.2.2, a modification of the WSum, called NWSum in Section 3.3.2.3 and the Sigma method in Section 3.3.2.4.

3.3.2.1 Crowding Distance

The Crowding Distance (CD) metric is calculated following the procedure previously presented in Section 3.3.1.1.

In the CD leader selection method, two leader candidates are randomly chosen and the one presenting higher CD is considered to be in a less crowded region, hence it is selected. This is the leader selection method originally used in SMPSO.

3.3.2.2 Weighted Sum

The Weighted Sum approach is introduced by Branke and Mostaghim (2006), and consists of a weighted sum of the objective values. The value is calculated according to Equation (3.3). Where u_j is the j-th value of the objective vector $\boldsymbol{u}$, l_j is the j-th objective value of the leader candidate $\boldsymbol{l}$ and m is the number of objectives. It is noted that higher values of WSum are attributed to better objective values (for convex maximization problems).

$$F = \sum_{j=1}^{m} \frac{u_j}{\sum_{k=1}^{m} u_k} l_j \tag{3.3}$$

The leader candidate that obtains the smallest weighted sum is selected to lead the particle $\boldsymbol{u}$. In this method, the leader selected will be the closest to the opposite axis to the particle $\boldsymbol{u}$ (in convex problems), hence this method introduces diversity in the search. Its general behavior is shown in Figure 3.7.

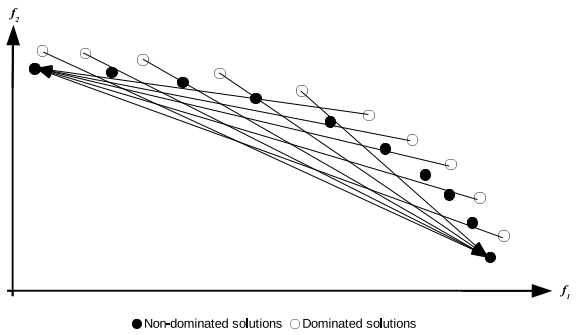


Figure 3.7: Representation of the method WSum

In this figure, the black points represent leader candidates within the repository, the white points represent the particles in the search and the arrows point toward the leader that a particle would choose during the search for a bi-objective minimization problem. In this problem is possible to note that the particles would select leaders close to the axis, but farther from their current position.

3.3.2.3 NWSum

The NWSum method was proposed by Padhye et al. (2009) and is a variation of the method WSum previously presented. In WSum, the solution with the smallest weighted sum is selected as leader, however in NWSum the solution with the largest weighted sum is selected instead.

Figure 3.8 shows the behavior of the method NWSum, where the black points represent the leader candidates within the repository, the white points represent the particles of the search

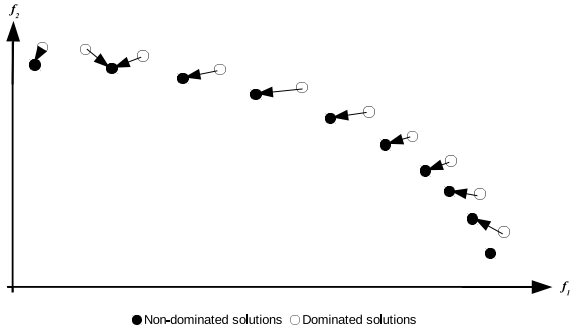


Figure 3.8: Representation of the method NWSum

and the arrows point toward the leader that would be selected by a particle during the search for a bi-objective minimization problem. In this example, each particle selects a leader close to it, this behavior improves the convergence.

3.3.2.4 Sigma

In the Sigma method (Mostaghim and Teich, 2003), the leader for a particle is selected according to its Sigma distance. For each solution $\boldsymbol{u}$ in both, the repository and the swarm, a Sigma vector is calculated. In some cases the Sigma vector can present negative values without compromising its performance since the Euclidean distance between the vectors are calculated. For a bi-objective problem, the Sigma vector is defined as follows:

$$\boldsymbol{u_\sigma} = \frac{u_1^2 - u_2^2}{u_1^2 + u_2^2} \tag{3.4}$$

For problems with more than two objectives, the Sigma is a vector with $\binom{m}{2}$ elements, representing the combinations of Equation (3.4) for all the objectives. In this method, a particle selects as leader the solution whose Sigma vector is closer to its own considering the Euclidean distance.

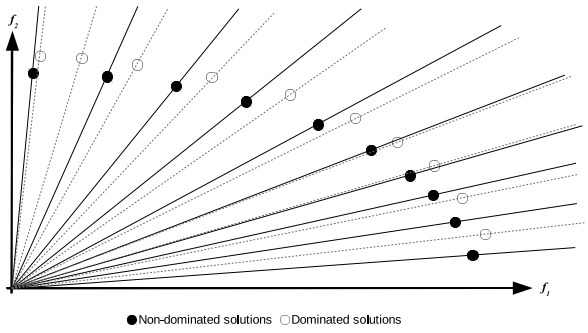


Figure 3.9: Representation of the method Sigma.

Figure 3.9 shows the behavior of the Sigma method, where the black points represent leader candidates within the repository and the white points represent the particles in the search in a bi-objective minimization problem. The Sigma vector of both the leader candidates and the particles are calculated, and represented by the black solid lines, and the gray dotted lines respectively. These vectors start from the origin and cross the solutions. Each particle, in this case, selects the leader candidate whose Sigma vector is closer to its own.

3.3.3 SMPSO

By studying the inability of some state-of-the-art MOPSOs in satisfactorily solving certain multi-modal problems, it was found by Nebro et al. (2009) that the velocity of the particles in these algorithms can become too high, resulting in erratic movements toward the limits of the positions of the particles. This behavior is known as "swarm explosion" (Clerc and Kennedy, 2002) and can be prevented by using a velocity constriction mechanism.

Thus, taking OMOPSO (Sierra and Coello, 2005) as base algorithm, Nebro et al. (2009) proposed the Speed-constrained Multi-objective PSO (SMPSO), which incorporates a velocity constriction procedure.

To control the velocity of the particles, instead of using upper and lower parameter values, it is adopted a constriction coefficient calculated according to Equation (3.5) obtained from the constriction factor χ originally presented by Clerc and Kennedy (2002).

$$\chi = \frac{2}{2 - \varphi - \sqrt{\varphi^2 - 4\varphi}} \tag{3.5}$$

where

$$\varphi = \begin{cases} C_1 + C_2 & \text{if} \quad C_1 + C_2 > 4 \\ 1 & \text{if} \quad C_1 + C_2 \leq 4 \end{cases} \tag{3.6}$$

However it has been noted that in the case $\varphi = 1$, then $\sqrt{1^2 - 4 \times 1} = \sqrt{-3}$. Due to this fact, in our implementation we replace Equation (3.6) by the version used by the SMPSO algorithm implemented in the jMetal framework (Durillo et al., 2010) and reproduced in Equation (3.7).

$$\begin{cases} \varphi = C_1 + C_2 & \text{if} \quad C_1 + C_2 > 4 \\ \chi = 1 & \text{if} \quad C_1 + C_2 \leq 4 \end{cases} \tag{3.7}$$

where in case $C_1 + C_2 > 4$, χ is calculated by Equation (3.5), but in the case $C_1 + C_2 \leq 4$, the value of χ is 1, Equation (3.5) is not used, and the velocity of the particle is not affected.

The final velocity of the particle i is defined according to Equation (3.8)

$$\boldsymbol{v}_i^{(t)} = \chi \times (\omega \times \boldsymbol{v}_i^{(t-1)} + C_1 \times r_1 \times (\boldsymbol{x}_{b_i} - \boldsymbol{x}_i^{(t-1)}) + C_2 \times r_2 \times (\boldsymbol{x}_{g_i} - \boldsymbol{x}_i^{(t-1)}) \tag{3.8}$$

Besides, a mechanism is created where the accumulated velocity of each variable j of each particle is pruned through Equation (3.9).

$$v_{i,j}^{(t)} = \begin{cases} \Delta_j & \text{if} \quad v_{i,j}(t) > \Delta_j \\ -\Delta_j & \text{if} \quad v_{i,j}(t) \leq -\Delta_j \\ v_{i,j}(t) & \text{otherwise} \end{cases} \tag{3.9}$$

where

$$\Delta_j = \frac{(ub_j - lb_j)}{2} \tag{3.10}$$

and $\boldsymbol{ub}$, $\boldsymbol{lb}$ represent the upper and lower bounds of the decision variables of the optimization problem respectively.

Summarizing the procedure, the velocity of the particles is calculated according to Equation (3.8), which uses the constriction factor previously calculated. Then the velocity is constrained by Equation (3.9) resulting in the final velocity used to calculate the new position of the particles (Nebro et al., 2009).

3.4 MOEA/D

Multi-objective evolutionary algorithm based on decomposition (MOEA/D) is a representative algorithm of the decomposition-based method, proposed by Zhang and Li (2007). MOEA/D is characterized by decomposing a multi-objective problem into several single-objective subproblems, each one defined by a weight vector $\boldsymbol{w}^i = (w_1^i, ..., w_m^i)$, and solving them cooperatively by defining a neighborhood relation among them. MOEA/D has two major features: local mating and local replacement. Local mating means that the mating parents are usually selected from neighboring subproblems. Local replacement means that an offspring is usually compared with solutions of neighboring subproblems. However, as discussed by Li and Zhang (2009) it is helpful for the population diversity to conduct mating and replacement from all subproblems with a certain (low) probability (Li et al., 2015). The MOEA/D framework is depicted on Algorithm 5.

Algorithm 5: MOEA/D

Input: set of weight vectors $\boldsymbol{W} = \{\boldsymbol{w}^1, ..., \boldsymbol{w}^{|W|}\}$, neighborhood size T.
Output: population $\boldsymbol{P}$.

```
[P, B] = initialize() // Parent population P, neighborhood index set B
while t < t_max do
    for i = 1, ..., |W| do
        P̄ = selection(B_i, P) // Randomly selects two parents from the neighborhood
        O = reproduction(P̄) // generates new solution(s) by using genetic operators
        O' = mutation(O) // apply a mutation operator
        for j ∈ B_i do
            x' = min_{y∈O'} g(y|w^j, z) // selects the best solution from O' regarding w^j
            x^j = min(x^j, x')
        end
    end
    t = t + 1
end
return P = {x^1, ..., x^|W|}
```

The MOEA/D algorithm works as follows: first the population $\boldsymbol{P}$ is randomly initialized with $|\boldsymbol{W}|$ random solutions, and each solution is associated with a subproblem. Then the neighborhood $\boldsymbol{B}$ is initialized, with $\boldsymbol{B}_i = \{i_1, ..., i_T\}$, where $\{\boldsymbol{w}^{i_1}, ..., \boldsymbol{w}^{i_T}\}$ are the T closest weight vectors to $\boldsymbol{w}^i$ (considering the Euclidean distance). Next, for a predefined number of generations, and for each subproblem (defined by a weight vector) two random solutions are selected from its neighborhood, note that with a predefined probability $(1 - \delta)$ (usually small) the solutions might be selected from the whole population. Following, one or more solutions are generated using genetic operators, then these solutions are usually mutated. The final operation for a subproblem is the update, where each of its neighbors j (or the entire population with probability $1 - \delta$), selects the best solution $\boldsymbol{x}'$ (regarding its weight vector $\boldsymbol{w}^j$), and, if this solution is better than its current best solution $\boldsymbol{x}^j$, then $\boldsymbol{x}'$ replace $\boldsymbol{x}^j$, note that a maximum number of replacements can be applied. At the end of the execution, the current population $\boldsymbol{P}$ is returned as output of the algorithm.

Any scalarizing function $g()$ can be used within MOEA/D. Usually the most used are weighted sum (WSum) (Zhang and Li, 2007), penalty-based boundary intersection (PBI) (Zhang and Li, 2007) and weighted Tchebycheff (TCH) (Zhang and Li, 2007). They are presented in Equations 3.11, 3.12 and 3.13 respectively, where $\boldsymbol{u}$ is the objective vector of a given solution, $\boldsymbol{w}$ is a weight vector and $\boldsymbol{z}^*$ is the ideal point, or the vector containing the best value found so far for each objective.

$$g^{WSum}(\boldsymbol{u}|\boldsymbol{w}) = \sum_{j=1}^{m}(w_j \times u_j) \tag{3.11}$$

$$g^{tch}(\boldsymbol{u}|\boldsymbol{w}, \boldsymbol{z}^*) = \max_{1 \leq j \leq m}\{|u_j - z_j^*| \times w_j\} \tag{3.12}$$

$$g^{pbi}(\boldsymbol{u}|\boldsymbol{w}, \boldsymbol{z}^*) = d_1 + \theta \times d_2 \tag{3.13}$$

where

$$d_1 = \frac{||(\boldsymbol{u} - \boldsymbol{z}^*)^T \times \boldsymbol{w}||}{||\boldsymbol{w}||} \text{ and } d_2 = ||\boldsymbol{u} - \left(\boldsymbol{z}^* + d_1 \times \frac{\boldsymbol{w}}{||\boldsymbol{w}||}\right)||.$$

The value θ in Eq 3.13 is a constant usually set to 5.0.

3.5 MOEA/DD

MOEA/DD is a state-of-the-art many-objective optimizer proposed by Li et al. (2015). MOEA/DD employs dominance and decomposition concepts simultaneously and is based on the well-known MOEA/D (Zhang and Li, 2007). The main difference between MOEA/D and MOEA/DD is in their replacement mechanism. MOEA/D employs a simple replacement mechanism, where the best solution for each subproblem can only be replaced by a new solution if it has a better scalarizing value than the former. MOEA/DD introduces a more complex replacement mechanism that combines dominance and decomposition principles in order to enhance the diversity characteristic of the algorithm. This intricate update mechanism is presented in Algorithm 6.

In this update procedure, one offspring $\boldsymbol{x}^c$ is considered each time. First the subproblem $\boldsymbol{\Phi}^c$ closest to $\boldsymbol{x}^c$ is identified through Equation (3.14),

$$\boldsymbol{\Phi}^c = \{\boldsymbol{u} \in \mathbb{R}^m | \langle \boldsymbol{u}, \boldsymbol{w}^c \rangle \leq \langle \boldsymbol{u}, \boldsymbol{w}^j \rangle\} \tag{3.14}$$

Algorithm 6: MOEA/DD-Update

Input: parent population $\boldsymbol{P}$, offspring solution $\boldsymbol{x}^c$.
Output: updated parent population $\boldsymbol{P}$.

Find the subproblem closest to $\boldsymbol{x}^c$ according to (3.14)
$\boldsymbol{P}' = \boldsymbol{P} \cup \boldsymbol{x}^c$
Calculate the dominance level of each solution in $\boldsymbol{P}'$
if $l = 1$ **then** // All solutions in $\boldsymbol{P}'$ are nondominated
    $\boldsymbol{x}'$ =locateWorst($\boldsymbol{P}'$)
    $\boldsymbol{P} = \boldsymbol{P}' \backslash \boldsymbol{x}'$
else
    if $|\boldsymbol{F}_l| = 1$ **then** // $\boldsymbol{F}_l$ has only one solution $\boldsymbol{x}^l$
        if $|\Phi^l| > 1$ **then** // $\boldsymbol{x}^l$ is associated with subproblem Φ^l
            $\boldsymbol{P} = \boldsymbol{P}' \backslash \boldsymbol{x}^l$
        else // $|\Phi^l| = 1$
            $\boldsymbol{x}'$=locateWorst($\boldsymbol{P}'$)
            $\boldsymbol{P} = \boldsymbol{P}' \backslash \boldsymbol{x}'$
        end
    else
        Identify the most crowded subproblem Φ^h associated with those solutions in $\boldsymbol{F}_l$
        if $|\Phi^h| > 1$ **then**
            Find the worst solution
            $\boldsymbol{x}' = \text{argmax}_{\boldsymbol{x} \in \Phi^h} g(\boldsymbol{u}' | \boldsymbol{w}^h, \boldsymbol{z}^*)$
            $\boldsymbol{P} = \boldsymbol{P}' \backslash \boldsymbol{x}'$
        else // $|\Phi^h| = 1$
            $\boldsymbol{x}'$=locateWorst($\boldsymbol{P}'$)
            $\boldsymbol{P} = \boldsymbol{P}' \backslash \boldsymbol{x}'$
        end
    end
end
return $\boldsymbol{P}$

where $j \in \{1, ..., |\boldsymbol{W}|\}$, $|\boldsymbol{W}|$ is the number of subproblems, $\boldsymbol{u} = \boldsymbol{f}(\boldsymbol{x}) \in \Omega$, and $\langle \boldsymbol{u}, \boldsymbol{w}^j \rangle$ is the acute angle between $\boldsymbol{u}$ and $\boldsymbol{w}^j$. Then $\boldsymbol{x}^c$ is combined with $\boldsymbol{P}$, forming a hybrid population $\boldsymbol{P}'$. Next the dominance level structure (l) of $\boldsymbol{P}'$ is calculated, as in NSGA-II (Deb et al., 2000) and one of the following two scenarios arise.

1. There is only one nondomination level ($l = 1$): Since all members in $\boldsymbol{P}'$ are nondominated from each other, different measures have to be used such as density estimation and scalarization function to distinguish between solutions. In this case, Algorithm 7 is used to locate the worst solution of the most crowded subproblem.

2. There are more than one nondomination levels ($l > 1$): Since only one solution needs to be eliminated from $\boldsymbol{P}'$, the decision process starts from the last nondomination level $\boldsymbol{F}_l$. Then, there are two cases.

    2.1 $\boldsymbol{F}_l$ contains only one solution $\boldsymbol{x}^l$($|\boldsymbol{F}_l| = 1$). First of all, we investigate the density of the subproblem (Φ^l) associated with $\boldsymbol{x}^l$.

Algorithm 7: locateWorst

Input: hybrid population $\boldsymbol{P'}$.
Output: the worst solution $\boldsymbol{x'}$.
1 Identify the most crowded subproblem Φ^h in $\boldsymbol{P'}$, tie is broken as $h = \text{argmax}_{i \in S} \sum_{\boldsymbol{x} \in \Phi^i} g(\boldsymbol{u}|\boldsymbol{w}^i, \boldsymbol{z}^*)$
2 In Φ^h, find the solution set $\boldsymbol{R}$ that belongs to the worst nondomination level
3 Find the worst solution $\boldsymbol{x'} = \text{argmax}_{\boldsymbol{x} \in \boldsymbol{R}} g(\boldsymbol{u}|\boldsymbol{w}^h, \boldsymbol{z}^*)$
4 **return** $\boldsymbol{x'}$

- If Φ^l has more than one solution (including $\boldsymbol{x}^l$), $\boldsymbol{x}^l$ is removed from $\boldsymbol{P'}$. This is because $\boldsymbol{x}^l$ belongs to the last nondomination level and Φ^l has other better solutions.
- Otherwise, Φ^l is an isolated subproblem, in this case $\boldsymbol{x}^l$ is important for population diversity and should be kept without reservation. Instead we use Algorithm 7 to identify the worst solution of the most crowded subproblem.

2.2 $\boldsymbol{F_l}$ contains more than one solution ($|\boldsymbol{F_l}| > 1$). First we identify the most crowded subproblem Φ^h (tie is broken according to the sum of the scalarized values, largest is worst for minimization problems) associated with those solutions in $\boldsymbol{F_l}$.

- If Φ^h has more than one solution, we eliminate the worst solution $\boldsymbol{x'} \in \Phi^h$, which owns the largest scalarized value from $\boldsymbol{P'}$.
- Otherwise, if the niche count of Φ^h is one, it means that every member in $\boldsymbol{F_l}$ is associated with an isolated subproblem. As discussed before, such solutions should be preserved without reservation. In this case we use Algorithm 7 to identify the worst solution of the most crowded subproblem.

In Algorithm 6, when we cannot easily determine a solution to remove, we use Algorithm 7 to identify the worst solution of the most crowded subproblem. To estimate the density of a subproblem, we have just to count the number of solutions associated with it, where the most crowded subproblem Φ^h has the largest niche count. If more than one solution has the same largest niche count, we select the one with the largest sum of scalarized values. Then we identify the solutions in Φ^h that have the worst nondomination level, and among them, the solution that has the largest scalarized value. Thereafter $\boldsymbol{x'}$ is eliminated from $\boldsymbol{P'}$.

Based on the above update mechanism, MOEA/DD works as follows: first the algorithm is initialized, including loading the weight vectors, defining the neighborhood relation and randomly initializing $|\boldsymbol{W}|$ solutions, one for each subproblem. Then the solutions are evaluated and the ideal point is calculated. In the next step, until a stopping condition is met, the reproduction procedure is carried out, where, for each subproblem, two parents are randomly selected usually from the neighborhood and the simulated binary crossover (SBX) (Zhang and Li, 2007) and polynomial mutation are used to generate two offspring solutions. Following, the previously described update procedure is applied for each offspring (Following Algorithm 6). When the stopping condition is met, the population is returned as output of the algorithm.

3.6 EDA

Several types of Evolutionary Algorithms (EAs) have been proposed over the last few decades (Hauschild and Pelikan, 2011) and they follow more or less the same framework adopted

from natural evolution. Given a fitness function that evaluates the quality of solutions, the algorithm iteratively evolves a population of candidate solutions for the problem. New offspring solutions are reproduced from the fitter solutions of the population (survival of the fittest) by applying genetic operators (crossover and mutation) (Larrañaga et al., 2012).

The simplicity and good results in a variety of domains have brought a lot of attention and interest to these algorithms. A key of the success of EAs is the identification, preservation and effective combination of the fitter partial solutions of the problem during evolution (Harik et al., 1999). However, it has been shown that the traditional operators used in EAs fail to properly accomplish this task when certain characteristics are present in the problem (Pelikan, 2005). A main reason for this shortcoming is that these algorithms do not properly consider the dependencies and relationships between the variables of the problem, and are not able to thoroughly exploit the information obtained so far in order to speed up convergence. There are properties like non-linearity, ill-conditioning and deception in real world problems that without considering these types of regularities can pose significant challenges to traditional EAs (Larrañaga et al., 2012).

Probabilistic modeling can offer a systematic way of acquiring this kind of regularities, and therefore can help to achieve a quick, accurate and reliable problem solving. For this purpose, instead of genetic operators used in traditional EAs, in each iteration new candidate solutions of the problem are generated by estimating a probabilistic model based on the statistics collected from the set of candidate solutions and sampling the learnt probabilistic model (Larrañaga et al., 2012).

Therefore, the problem regularities encoded in the probabilistic model are used when generating new solutions, thus trying to overcome the shortcomings of traditional EAs. This incorporation of probabilistic modeling into EAs has led to a new paradigm called Estimation of Distribution Algorithms (EDAs) or competent EAs (Larrañaga and Lozano, 2002; Larrañaga et al., 2012; Santana et al., 2009).

EDAs assume that it is possible to model promising areas of the search space, and to use this model to guide the search for the optimal solutions. The model learnt in EDAs captures an abstract representation of the features shared by the selected solutions and encodes the different patterns of interactions between subsets of the problem variables. The advantage of EDAs over traditional EAs in dealing with the problems that contain important interactions among their variables, together with the capacity to solve different types of problems in a robust and scalable manner has greatly popularized these algorithms (Larrañaga et al., 2012).

General EDAs iterate three steps until some termination criterion is satisfied: select good solutions from a population, estimate the probability distribution from the selected individuals (learn a model) and generate new solutions (sample) from the estimated distributions (model). Figure 3.10 graphically represents this scheme (Ahn et al., 2006; Larrañaga et al., 2012).

A pseudocode of a general EDA framework is presented in Algorithm 8. In this algorithm, firstly a population $\boldsymbol{P}$ is randomly generated, then for a predetermined number of iterations (t_{max}) promising candidate solutions are selected from the whole population to be used in the learning phase to create a probabilistic model $\mathcal{M}$. A set of offspring solutions is sampled from this probabilistic model and the new population is updated based on the previous one and the set of new offspring solutions. At the end of the iterations, the best solution from the population is returned as result of the algorithm.

In spite of similar behavior patterns, EDAs can be characterized by the methods of learning a probabilistic model and sampling new solutions, and their performance is affected directly by the efficiency of these methods. In this work we are particularly interested in two EDAs from different classes, the first one is based on probabilistic graphical models and

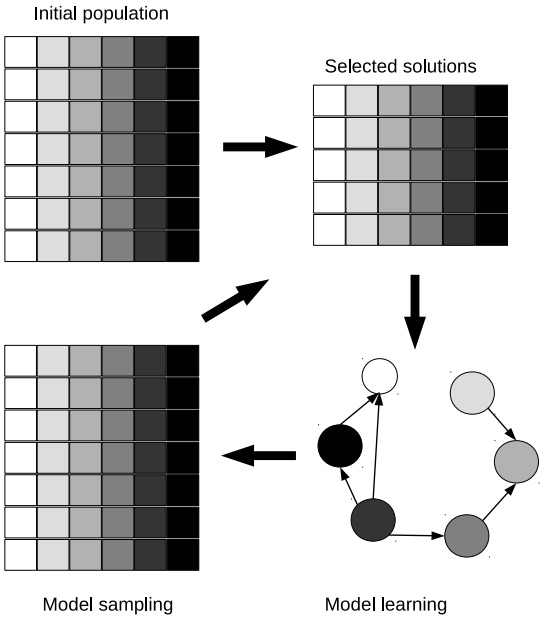


Figure 3.10: Representation of a general EDA.

called real-coded Bayesian optimization algorithm (rBOA) (Ahn et al., 2006). The second is based on Gaussian modeling and, named covariance matrix adaptation evolution strategy (CMA-ES) (Hansen and Ostermeier, 1996).

rBOA was selected because it is a continuous version of the Bayesian optimization algorithm (BOA) (Pelikan et al., 1999), which is one of the most efficient and extensively applied EDAs (Hauschild and Pelikan, 2011). Moreover, rBOA presents good results in the literature (Ahn et al., 2006), and has its source code is available in the web page of its creators[1]. The model learning and sampling of rBOA is presented in Section 3.6.1.

CMA-ES was selected to be used in this work because it is a state-of-the-art optimizer for single-objective functions, whose variant (Loshchilov, 2013) is the winner of the competition on real-parameter single objective optimization at CEC 2013. Moreover, CMA-ES has its source code available in the web page of its creators. The model learning and sampling of CMA-ES is described in Section 3.6.2.

3.6.1 rBOA

The real-coded Bayesian optimization algorithm (rBOA) (Ahn et al., 2006), is a continuous, real-coded version of the Bayesian optimization algorithm (BOA) (Pelikan et al., 1999). BOA is one of the most efficient and extensively applied EDAs (Hauschild and Pelikan, 2011), able to represent higher order dependencies between discrete variables by means of Bayesian networks.

Since rBOA is an extension of BOA to handle continuous variables by using mixture models, first the model learning of BOA is described, then we introduce the basic concepts of mixture distributions, following, we present how these two components are merged together to create rBOA.

rBOA combines concepts and methods from different areas, in particular.

- Bayesian networks are used to model the dependencies between the variables.

[1]http://parallel.kjist.ac.kr/ cwan/

Algorithm 8: General EDA framework

```
   // Step 1: Initialization
1  P=generateRandomPopulation()
2  for t = 0 < t_max do
       // Step 2: Selection
3      P̄=selectPromisingCandidates(P)
       // Step 3: Learn
4      M=learnProbabilisticModel(P̄)
       // Step 4:
5      O=generateOffspring(M)
       // Step 5:
6      P=generateNewPopulation(O,P)
7  end
8  return bestFrom(P)
```

- Model learning is similar, but not identical to the one used by BOA, using a score-search method.
- A mixture of normal distributions is used to model the relationships between the continuous variables.

In the following we explain these characteristics in detail and comment on other aspects relevant to rBOA.

3.6.1.1 Bayesian networks

Both BOA and rBOA rely on Bayesian networks (Pearl, 1988), which are defined by two components (Larrañaga et al., 2012; Pelikan, 2002):

- **Structure** ζ, which is defined by a directed acyclic graph with the nodes corresponding to the variables in the modeled data set and the edges corresponding conditional dependencies.
- **Parameters** θ, which contain, for each variable, the conditional probability distribution of its values given different value settings for its parents, according to the structure.

Mathematically, a Bayesian network encodes a joint probability distribution given by:

$$p(\boldsymbol{X}) = \prod_{i=1}^{n} p(X_i|\Pi_i) \tag{3.15}$$

where $\boldsymbol{X} = (X_1, ..., X_n)$ is a vector of all the variables in the problem, whose instantiations are denoted by $\boldsymbol{x} = (x_1, ..., x_n)$ (x_i denotes a possible value of X_i); Π_i is the set of parents of X_i in the network (the set of nodes from which there exists an edge to X_i); and $p(X_i|\Pi_i)$ is the conditional probability of X_i given its parents Π_i (Pelikan, 2002).

A directed edge relates the variables so that in the encoded distribution, the variable corresponding to the terminal node is conditioned on the variable corresponding to the initial node.

An example of Bayesian network is presented by Russell and Norvig (2011) and a simplified version is reproduced in Figure 3.11. In the structure of the network, conditional dependencies are encoded, for instance, the alarm ring depends on whether there is a burglar on the house, or there is an earthquake. Moreover, conditional independencies are assumed, for instance, the earthquake is independent of whether an alarm is installed or not. To fully specify the Bayesian network, tables of conditional probabilities for each variable are present, where the letters B, E and A means Burglary, Earthquake and Alarm respectively.

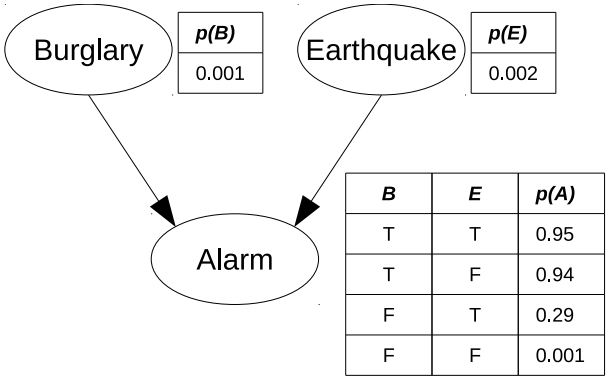


Figure 3.11: An example of Bayesian network with conditional probabilities for each variable.

3.6.1.2 Model learning of BOA

In order to learn a network that reflects the dependencies and independencies that properly decompose a problem, BOA conducts two subtasks (Pelikan, 2002):

- **Learning the structure, or model selection**. First, the structure of a network must be determined. This structure defines the conditional dependencies and independencies encoded by the network.
- **Learning the conditional probabilities, or model fitting**. The structure also identifies conditional probabilities that must be specified for a complete model. After learning the structure, the values of the conditional probabilities with respect to the final structure must be learned.

Learning the parameters is straightforward, because the value of each variable in the population of promising solutions is specified (i.e., the data is complete). To maximize the likelihood of the model with a fixed structure and complete data, the probabilities should be set according to the relative frequencies observed in the data. Thus, the parameters can be learned by iterating through all selected solutions and computing relative frequencies of different partial solutions (Pelikan, 2002).

Learning the structure is a much harder problem. The algorithms for learning the structure of Bayesian networks have two components (Pelikan, 2002; Pelikan et al., 2006):

1. **A scoring metric.** A scoring metric measures the quality of Bayesian network structures. Usually a scoring metric is proportional to the likelihood of the structure or it is equal to a combination of the likelihood and some penalty for complex models. However, other measures can be used, such as statistical tests on independence.

2. **A search procedure.** A search procedure searches the space of all possible network structures to find the best network with respect to a given scoring metric. The space of network structures can be restricted according to a bound on the complexity of networks or some other prior problem-specific knowledge.

In BOA, a penalized maximum-likelihood criterion known as the Bayesian Information Criterion (BIC) (Schwarz, 1978) is employed as scoring metric. Detailing the working of BIC is out of the scope of this thesis, however the interested reader is referred to (Ahn et al., 2006; Pelikan, 2002; Pelikan et al., 2006; Schwarz, 1978) for additional information.

BOA uses a simple greedy search algorithm to construct a network that maximizes the scoring metric. The greedy algorithm starts with an empty graph and proceeds by incrementally adding an edge (such that no cycles are introduced) that maximally improves the metric until no more improvement is possible (Ahn et al., 2006; Pelikan, 2002; Pelikan et al., 2006).

3.6.1.3 Mixture of normal distributions to model dependencies in rBOA

The probability density function (PDF) of a normal distribution is centered around its mean and decreases exponentially with the distance from the mean. If there are multiple subsets of values, a normal distribution must either focus on only one of these subsets, or it can embrace more than one subset at the expense of including the area between these subsets. In both cases, the resulting PDF cannot model the data accurately. One alternative to extend normal distributions to enable the coverage of multiple groups of similar points is to use a *mixture of normal distributions*. Each component of a mixture of normal distributions is a normal distribution itself and its called a *mixture component*. A coefficient, called *mixing coefficient* (β_i) is specified for each mixture component to denote the probability of a random point of belonging to this component. The PDF of a mixture is thus computed by multiplying the density function of each mixture component by the probability that a random point belongs to the component, and adding these weighted densities together (Bosman, 2003; Pelikan, 2002).

One possible way of estimating a mixture distribution from data is by using clustering. Clusters are possibly overlapping subvectors of the original sample vector such that each sample point in the original sample vectors occurs in at least one cluster. If the subvectors are mutually disjoint, we have *partitions* instead of clusters. The use of clusters allows to efficiently break up non-linear interactions. Hence, by estimating simpler probability distributions, such as factorizations[2], for each cluster separately, these probability distributions can be added into a mixture probability distribution to get an adequate representation of the complete sample vector. After clustering and estimating the simpler probability distributions for each cluster, we still have to choose the mixing coefficients. One of the common approaches if clusters are used, is to set β_i to the proportion of the size of the ith cluster with respect to the sum of the sizes of all clusters (Bosman, 2003).

3.6.1.4 Main steps of the rBOA algorithm

rBOA is able to learn complex dependencies between variables and make use of mixture models (McLachlan and Peel, 2000) to sample new solutions respecting the building blocks[3].

To achieve this task, first rBOA creates a Bayesian network as a product of conditionally independent distributions, accurately estimated based on subproblems (i.e., substructures).

[2]Decomposition of an object into a product of other objects, which when multiplied together give the original.

[3]Partial solutions that must not be disrupted.

Following, mixture models are employed for independently fitting each of these substructures. Subsequently, an independent substructure-wise sampling draws the offspring.

Model selection: As in discrete BOA, the model learning of rBOA is composed of a scoring metric and a search procedure. However, the BIC metric employed needs to be changed to cope with the use of mixture models. These mixture models are used for modeling the selected individuals by a mixture of probability distributions. Hence p($\boldsymbol{X}$) can be described by a linear combination of a number of mixture components.

Model fitting: The model fitting of rBOA must conduct the probability distribution of a problem as a product of conditionally independent distributions accurately estimated based on subproblems (i.e., subspace-based model fitting). However, unlike discrete EDAs, a preprocessing step is needed for explicitly discovering subproblems, before performing the subspace-based model fitting. This is because discrete EDAs can implicitly carry out the problem decomposition in the course of model learning, while real-coded EDAs cannot do so (Ahn et al., 2006).

In principle, the problem decomposition can be accomplished by discovering component subproblems of a decomposable problem. The component subproblem is defined by the set of a node and its parent in the Bayesian factorization graph. Another observation is that the set of a parent and its child nodes can be grouped as a kind of subproblem because the child nodes share a common feature even though they do not directly interact with each other. This set is denoted as the dual component subproblem (Ahn, 2006).

Since the fitting process must be applied to every component subproblem, which could be computationally expensive, it is not adequate to use directly the component subproblems for model fitting. An alternative is to obtain subproblems by discovering minimal compound subproblems, which are defined as the largest set in the component and dual component subproblems. There is another decomposition that is simple but quite efficient for large problems. Nodes in a maximally connected subgraph are looked on as a family as they have a common feature of being bound with common ancestors or descendants. Thus, the nodes can be thought of as interacting with each other in some sense. The conditional distributions can then be obtained from the probability distributions fitted over the maximally connected subgraphs, which are called maximal compound subproblems, without unduly compromising on the fitting accuracy (Ahn and Ramakrishna, 2008).

Subspace based model fitting: Following, each substructure must be independently fitted. Mixture models are employed as an efficient tool for this purpose. Its aim is twofold: comprehending the type of dependency between variables and traversing the search space effectively. Higher order factorized probability distributions are effective in discovering linear interactions between variables. Each mixture component can model the linearity of the variables. The mixture model can approximate any type of dependency with a combination of piecewise linear interaction models. In addition it has the effect of partitioning each subspace for effective search. Since Bayesian factorization and mixture models are being employed, $P(\boldsymbol{X})$ can be written as a product of linear combinations of subspace-based PDFs (Ahn and Ramakrishna, 2008).

Model sampling: After model fitting, new individuals are generated from sampling the resulting Bayesian network. Due to its simplicity and efficiency, probabilistic logic sampling (PLS) (Henrion, 1986) is employed, based on the PDFs of the mixture components of each subproblem.

3.6.2 CMA-ES

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (Hansen and Ostermeier, 1996) combines characteristics of both evolution strategies (Beyer and Schwefel, 2002) and EDAs. It is considered a state-of-the-art optimizer for single-objective continuous functions.

One of the distinguishing features of CMA-ES is the use of a covariance matrix to estimate the probability of the moves that lead to more promising solutions. This matrix is updated throughout the evolution in a process that injects information from the best solutions found so far into the matrix.

CMA-ES works by sampling solutions from a multi-variate normal distribution based on a mean vector $\bar{\boldsymbol{m}}$, an $n \times n$ covariance matrix $\mathbf{C}$, and a step size σ. The search progresses by iteratively adapting these parameters in order to obtain better solutions. To accomplish this, the generated solutions are ranked based on their quality, originally determined by their fitness. Then, these ranked solutions are used in weighted equations to update the parameters of the distribution for the next generation.

The CMA-ES starts with the mean being initialized in the mean of the search space $[lb, ub]$ as $\bar{\boldsymbol{m}} = (lb + \frac{ub-lb}{2}, ..., lb + \frac{ub-lb}{2})$. The covariance matrix $\mathbf{C}$ is usually initialized as the identity matrix. The evolution paths used to update the step size ($\boldsymbol{p_\sigma}$) and the covariance matrix ($\boldsymbol{p_c}$) are set to $\boldsymbol{p_\sigma} = \boldsymbol{p_c} = (0, ..., 0)$, and the step size is $\sigma = \frac{ub-lb}{4}$.

In an iteration t of CMA-ES, the following operations are performed:

1. Sample a set of $\lambda \geq 1$ solutions from the multivariate normal distribution following Equation (3.16):

$$\boldsymbol{x}_i^{t+1} \sim \mathcal{N}\left(\bar{\boldsymbol{m}}^t, \left(\sigma^t\right)^2 \mathbf{C}^t\right), i = 1, \ldots, \lambda \tag{3.16}$$

2. Evaluate the solutions and order them according to their quality such that $\boldsymbol{x}_{i:\lambda}$ represents the i-th best solution.

3. Calculate the new mean vector ($\bar{\boldsymbol{m}}$) through a weighted recombination of the μ best solutions as in Equation (3.17), where the weights are usually defined linearly or logarithmicaly decreasing:

$$\bar{\boldsymbol{m}}^{t+1} = \sum_{i=1}^{\mu} w_i \boldsymbol{x}_{i:\lambda}^{t+1}, \sum_{i=1}^{\mu} w_i = 1, w_i > 0 \tag{3.17}$$

4. Update the evolution path $\boldsymbol{p}_\sigma$ through Equation (3.18):

$$\boldsymbol{p_\sigma} = (1 - c_\sigma)\boldsymbol{p_\sigma} + \sqrt{c_\sigma(2 - c_\sigma)\mu_{eff}} \times (\mathbf{C}^t)^{-\frac{1}{2}} \times \frac{\bar{\boldsymbol{m}}^{t+1} - \bar{\boldsymbol{m}}^t}{\sigma^t} \tag{3.18}$$

   where the variance effective selection mass is $\mu_{eff} = \frac{1}{\sum_{i=1}^{\mu} w_i^2}$, $\frac{1}{c_\sigma}$ is the backward time horizon of the evolution path $\boldsymbol{p}_\sigma$, set as $c_\sigma = \frac{\mu_{eff}+2}{n+\mu_{eff}+5}$.

5. Based on the previously calculated evolution path, update the step size σ as shown in Equation (3.19):

$$\sigma^{t+1} = \sigma^t \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{||\boldsymbol{p_\sigma}||}{E||\mathcal{N}(0, I)||} - 1\right)\right) \tag{3.19}$$

where the damping parameter is set as $d_\sigma = 1 + 2\max(0, \sqrt{\frac{\mu_{eff}-1}{n+1}} - 1) + c_\sigma$, and the expectation of the norm of an $\mathcal{N}(0, I)$ random vector is set as $E||\mathcal{N}(0, I)|| \approx \sqrt{n}(1 - \frac{1}{4n} + \frac{1}{21n^2})$.

6. Update the evolution path $\boldsymbol{p}_c$ through Equation (3.20):

$$\boldsymbol{p_c} = (1 - c_c)\boldsymbol{p_c} + h_\sigma \sqrt{c_c(2 - c_c)\mu_{eff}} \times \frac{\bar{\boldsymbol{m}}^{t+1} - \bar{\boldsymbol{m}}^t}{\sigma^t} \tag{3.20}$$

where $\frac{1}{c_c}$ is the backward time horizon of the path $\boldsymbol{p}_c$ set as $c_c = \frac{4+\mu_{eff}/n}{n+4+2\mu_{eff}/n}$. The Heaviside function can stall the update of $\boldsymbol{p}_c$ if $||\boldsymbol{p_\sigma}||$ is large, preventing a too fast increase of axis of $\mathbf{C}$ in a linear surrounding (when the step size is far too small), it is defined as $h_\sigma = 1$ if $\frac{||\boldsymbol{p_\sigma}||}{\sqrt{1-(1-c_\sigma)^{2(t+1)}}} < (1.4 + \frac{2}{n+1})E||\mathcal{N}(0, I)||$, or 0 otherwise.

7. Finally, update the covariance matrix based on the previously calculated parameters through Equation (3.21):

$$\begin{aligned}\mathbf{C}^{t+1} =& (1 - c_1 - c_\mu)\mathbf{C}^t + c_1 \left(\boldsymbol{p_c}\boldsymbol{p_c}^T + \delta(h_\sigma)\mathbf{C}^t\right) + \\ & c_\mu \sum_{i=1}^{\mu} w_i OP\left(\frac{\boldsymbol{x}_{i:\lambda}^{(t+1)} - \bar{\boldsymbol{m}}^t}{\sigma^t}\right)\end{aligned} \tag{3.21}$$

where the learning rate for the rank-one update of the covariance matrix update is $c_1 = \frac{2}{(n+1.3)^2+\mu_{eff}}$, the learning rate for the rank-μ update of the covariance matrix update is $c_\mu = \min\left(1 - c_1, \alpha_\mu \frac{\mu_{eff}-2+1/\mu_{eff}}{(n+2)^2+\alpha_\mu\mu_{eff}/2}\right)$ with $\alpha_\mu = 2$. $\delta(h_\sigma) = (1 - h_\sigma)c_c(2 - c_c) \leq 1$ is of minor relevance and can be set to 0. The outer product of a vector with itself is denoted as OP: $\mathbb{R}^n \to \mathbb{R}^{n\times n}, \boldsymbol{x} \mapsto \boldsymbol{x}\boldsymbol{x}^T$, which results in a matrix of rank one with eigenvector $\boldsymbol{x}$ and eigenvalue $||\boldsymbol{x}||^2$.

A drawback of the CMA-ES is that while the algorithm is robust to large irregularities in the objective function, even small changes in the update procedure can lead to a dramatic break down of its performance. A method for making CMA-ES robust to changes of the solutions used in the update procedure is proposed by Hansen (2011). This method allows to safely inject external solutions into the CMA-ES procedure. The only change needed to make CMA-ES handle injected solutions is to restrict the distance between the injected solution and the previous mean, and to rescale the corresponding search step accordingly before taking it into account in the updates of mean, step size and covariance matrix (Zapotecas-Martínez et al., 2015).

This rescaling can be accomplished by means of Equation (3.22)

$$\boldsymbol{y}_i = \alpha_{clip}(c_y, ||(\boldsymbol{C}^t)^{-\frac{1}{2}}\boldsymbol{y}_i||) \times \boldsymbol{y}_i \quad \text{if } \boldsymbol{x}_i \text{ was injected,} \tag{3.22}$$

where $\alpha_{clip}(c, x) = \min(1, \frac{c}{x})$, $c_y = \sqrt{n} + 2n/n + 2$ and $\boldsymbol{y}_i^{t+1} = \frac{\boldsymbol{x}_i^{t+1} - \bar{\boldsymbol{m}}^t}{\sigma^t}$.

To enable injection in the CMA-ES, we just need to introduce Equation (3.22) before Equation (3.17) in step 3, and slightly change Equation (3.19) to:

$$\sigma^{t+1} = \sigma^t \exp\left(\min\left(\Delta_\sigma^{max}, \frac{c_\sigma}{d_\sigma}\left(\frac{||\boldsymbol{p_\sigma}||}{E||\mathcal{N}(0, I)||} - 1\right)\right)\right) \tag{3.23}$$

where $\Delta_{\sigma}^{max} = 1$.

Another important addition to CMA-ES is to test for some reset criteria, as presented by Auger and Hansen (2005); Zapotecas-Martínez et al. (2015). These criteria are important to avoid stagnation and allow restarting a CMA-ES instance. They are described as follows:

- *NoEffectCoord.* Reset if adding 0.2-standard deviation in any coordinate does not change $\bar{\boldsymbol{m}}^t$.
- *NoEffectAxis.* Reset if adding a 0.1-standard deviation vector in any principal axis direction of $\boldsymbol{C}^t$ does not change $\bar{\boldsymbol{m}}^t$. More formally, restart if $\bar{\boldsymbol{m}}^t$ equals to $\bar{\boldsymbol{m}}^t + 0.1\sigma^t\sqrt{d_{jj}}\boldsymbol{b}_j$, where $j = (t \mod n) + 1$ and d_{jj} and $\boldsymbol{b}_j$ are respectively the j^{th} eigenvalue and eigenvector of $\boldsymbol{C}^t$, with $||\boldsymbol{b}_j|| = 1$.
- *TolXUp.* Reset if $\sigma^t \times \max(\text{diag}(\boldsymbol{D}))$ increased by more than 10^4.
- *ConditionCov.* Reset if the condition number of the covariance matrix exceeds 10^{14}.

3.7 Hyper-heuristics

The task of choosing appropriate parameters or algorithms to solve an optimization problem is often hard. Mainly because of the lack of guidance on how to select them and the low level of understanding on why different heuristics work effectively or not in different situations. The "No Free Lunch Theorem" proposed by Wolpert and Macready (1997) showed that, when averaged over all problems defined on a given finite search space, all search algorithms had the same average performance. This theorem helped to focus attention on the question of what sorts of problems any given algorithms might be particularly useful for (Burke et al., 2013; Glover, 2003).

Since different heuristic or components have different strengths and weaknesses, it makes sense to see whether they can be combined in some way so that each makes up for the weaknesses of another. In this context, the hyper-heuristic approach emerges as a high-level methodology which, when given a particular problem instance or class of instances, and a number of low-level heuristics or components, automatically produces an adequate combination of these components to effectively solve the given problem (Burke et al., 2013; Glover, 2003).

Hyper-heuristics operate at a higher level of abstraction than meta-heuristics and often have no knowledge of the domain. They only have access to a set of low-level heuristics that they can call upon, but with no knowledge of the purpose or function of a given low-level heuristic. This approach is proposed to allow a hyper-heuristic to operate in different problem domains by only replacing the set of low-level heuristics and evaluation functions (Glover, 2003).

Figure 3.12 shows a general hyper-heuristic framework. In the figure there is a barrier between the low-level heuristics and the hyper-heuristic. Domain knowledge is not allowed to cross this barrier. Therefore the hyper-heuristic has no knowledge of the domain under which it is operating. It only knows it has a set of low-level heuristics on which to call and it knows it will be passed the results of a given solution once it has been evaluated by the evaluation function (Glover, 2003).

According to Burke et al. (2013), hyper-heuristics can be classified by the nature of the heuristic search space as *heuristic selection*: methodologies for choosing or selecting existing heuristics, and *heuristic generation*: methodologies for generating new heuristics from the components of existing ones. Heuristic selection and generation can be further categorized as *perturbative*, which consider complete candidate solutions and change them by modifying

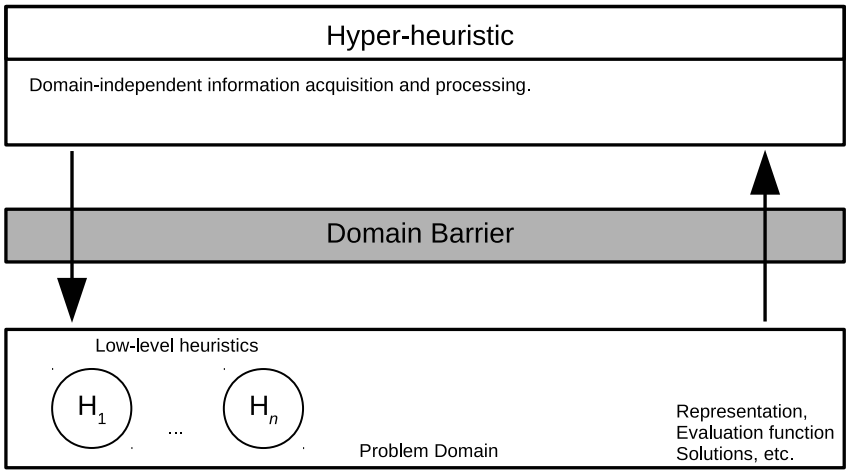


Figure 3.12: Representation of a general hyper-heuristic framework.

one or more of their solution components or *constructive*, where partial candidate solutions are considered, where one or more solution components are missing, and iteratively extending them.

Hyper-heuristics that use some feedback from the search process are considered learning algorithms and can be classified according to the source of the feedback as *online*: when the learning takes place while the algorithm is solving an instance of a problem, or *offline*: the knowledge is gathered in the form of rules or programs from a set of training instances that hopefully generalize to solving unseen instances (Burke et al., 2013).

In this work we are mostly interested in online perturbative heuristic selection methods. These techniques aim to improve a candidate solution by automatically selecting and applying a heuristic. Usually, a perturbative heuristic selection method combines two separate components: Heuristic selection and move acceptance. The *Heuristic selection* component is responsible for selecting and evaluating the low-level heuristics. There are different heuristic selection methods; the simplest are methods which have no learning mechanism, e.g. a random selection. There are also rank-based heuristic selection methods which use an update rule to rank the low-level heuristics based on their performance and where the best-ranked heuristics are more likely to be selected. Others use a more sophisticated method considering the history of performances obtained by the low-level heuristics. It is important to note that even simple methods are usually better than applying any low-level heuristic individually (Burke et al., 2013; Castro Jr. and Pozo, 2015). Among the most known selection methods are: Choice Function (Cowling et al., 2001; Maashi et al., 2014); Reinforcement learning; and Adaptive Operator Selection (Burke et al., 2013).

The *Move acceptance* component evaluates the solution generated by the applied heuristic using some quality information received from the problem domain, usually the fitness value of the solution. Then the solution is accepted or rejected depending on its performance and the move acceptance method rules. If a solution is better than a previous one it is accepted; otherwise an acceptance criterion is used. The use of an appropriated move acceptance method may increase the optimization performance substantially (Bilgin et al., 2007; Kheiri et al., 2016).

According to Burke et al. (2013) the decision of the move acceptance seems to be more important than the heuristic selection. This move acceptance strategy can be either deterministic or non-deterministic. *Deterministic* methods make the same acceptance decision regardless of the stage of the search using the current and new solution(s). *Non-deterministic* approaches might generate different decisions for the same input. Most non-deterministic move acceptance methods require additional parameters like the current iteration number.

In our study, we are interested in two heuristic selection methods that presented good results in the literature: Choice Function (Cowling et al., 2001) and Multi-armed Bandit (Fialho et al., 2010). These approaches are presented in Sections 3.7.1 and Section 3.7.2 respectively. Moreover, we propose a simple roulette wheel based heuristic selection method and present it in Section 3.7.3

3.7.1 Choice function

The choice function method is an online learning heuristic selection based on ranking, proposed by Cowling et al. (2001). The ranking is made based on a function that is the sum of three components (Ozcan et al., 2009) (f_1, f_2 and f_3), weighted by the scale factor parameters (α, β and γ).

$$cf(h_i) = \alpha f_1(h_i) + \beta f_2(h_j, h_i) + \gamma f_3(h_i) \tag{3.24}$$

1. The f_1 component is the quality of the low-level heuristic h_i. It is responsible for increasing the exploitation by raising the probability of the best performing heuristics.

2. f_2 is the quality of the pair of low-level heuristics (h_i, h_j) when applied together. The objective is to find a cooperative behavior between heuristics that performs well when applied together.

3. The function also has an exploration component (f_3). It computes the elapsed time since the last application of the low-level heuristic h_i. It increases the probability of the low-level heuristics not applied recently, to evaluate its current performance.

After the low-level heuristic is applied, the quality information is used to update the CF components. There are different ways of updating f_1 and f_2, for instance: using the last quality information received (Guizzo et al., 2015); or the accumulated quality information since the beginning of the search (Drake et al., 2012); or the mean value of the quality (Gonçalves et al., 2015b). For each heuristic i the f_2 component is updated considering the quality of i and the quality of the last applied heuristic j. The f_3 component can be computed using different ways as well, for instance: the elapsed time in seconds since the low-level heuristic i was applied, or the number of heuristic selections elapsed since the last time that i was applied.

To avoid the parameter configuration of α, β and γ, Drake et al. (2012) proposed an Adaptive Choice Function (ACF), which uses the following equation:

$$cf(h_i) = \phi f_1(h_i) + \phi f_2(h_j, h_i) + \gamma f_3(h_i) \tag{3.25}$$

Where ϕ is an exploitation factor for the best-performing heuristics, and γ is an exploration factor, responsible for increasing the probability of the heuristics that have not been recently applied. The parameter ϕ is set to 0.99 each time the heuristic improves the solution quality and decreased by 0.01 otherwise. The γ parameter is set to $(1 - \phi)$. The objective is to increase exploitation when the solution quality is improving and to increase exploration when the current best operators cannot increase the solution quality.

Gonçalves et al. (2015b) proposed two modifications: the first is the use of a scale factor (SF) parameter, since the measures used in f_1 and f_2 may be in different scales when compared to f_3. The second is the use of the mean values for f_1 and f_2 instead of the accumulated values, so the adaptation of the algorithm is faster.

3.7.2 Multi-armed Bandit

The Multi-armed Bandit is a problem that considers a set of A independent arms, with unknown probability of being chosen. The goal is to select the arms that maximize the accumulated reward along the time. The Multi-armed Bandit problem fits in the Exploitation vs. Exploration (EvE) dilemma.

Many algorithms have been proposed to tackle the MAB problem, one of them is the Upper Confidence Bound (UCB), which provides asymptotic optimality guarantees. In UCB, the selected arm is the one that maximizes the UCB function (Equation (3.26)) (Gonçalves et al., 2015a; Li et al., 2014).

$$\hat{q}_i + c \times \sqrt{\frac{2 \times \ln \sum_A n_{A,t}}{n_{i,t}}} \tag{3.26}$$

where, each arm has an empirically estimated quality ($\hat{q}_i$). Same way as the f_1 component from Choice Function; $\hat{q}_i$ is responsible for increasing exploitation, i.e., it enhances the probability of the best performing heuristics. The c parameter is a scale factor between exploration (the right term, that measures how frequently the arm has been tried) and exploitation (the left term, that measures the arm quality) (Gonçalves et al., 2015a; Li et al., 2014).

Some UCB-based algorithms have been used for Adaptive Operator Selection (AOS), as it aims to handle the EvE dilemma. The goal was to select the operators that increase the quality of the optimization solution output.

One of these algorithms is the Fitness-Rate-Rank-Based Multi-Armed Bandit (FRRMAB), proposed by Li et al. (2014). Usually, in UCB-based algorithms, the raw value of the fitness improvements is used as the reward. However, the range of these values may vary depending on the problem and the search stage. To deal with that, FRRMAB uses a fitness improvement rate (FIR), defined as Equation (3.27):

$$FIR = \frac{pf - cf}{pf} \tag{3.27}$$

Where the FIR of the operator is the difference between the fitness of the solution before (pf) and after (cf) applying the operator, divided by the old fitness (pf). Then, a sliding time window stores the FIR values of the last W^w applications. Moreover, the $Reward$ of an operator (i) is the sum of all FIR of that operator in the sliding window. Then, all operators are ranked by reward, and a decaying factor (D) is applied. D determines the priority level given to the best rewarded operator, a lower value will lead to a higher credit value for the top-ranked operators. The reward and decay are calculated through Equations (3.28) and (3.29) respectively:

$$Reward_i = \sum_{k \leftarrow 0}^{W^w} \begin{cases} FIR_k^{op} & \text{if } op = i \\ 0 & \text{otherwise} \end{cases} \tag{3.28}$$

$$Decay_i = D^{Rank_i} \times Reward_i \tag{3.29}$$

Finally, the credit value of the operator i is computed as Equation (3.30):

$$FRR_i = \frac{Decay_i}{\sum_{j=1}^{K} Decay_j} \tag{3.30}$$

To select the operators, the empirical reward $\hat{q}_i$ is replaced by the FRR_i value in the UCB function. Moreover, n_i indicates the number of times that the operator (i) has been applied in the last W^w iterations.

3.7.3 Roulette

The roulette based selection mechanism consists of a roulette wheel where the probability of each low-level heuristic is updated according to the difference in the quality value obtained before and after the application of the selected heuristic. The Roulette wheel starts with equal probability of choosing any of the available low-level heuristics. The Roulette is updated at each iteration based on the quality information. In a given iteration, if the selected heuristic improves or maintains the quality, then the probability of the selected heuristic is increased in the Roulette, the probabilities of the other heuristics are decreased, and the solution is accepted. However, if the quality decreases, the heuristic has its probability decreased, while the probabilities of the others are increased. This simple hyper-heuristic rewards or punishes the heuristics by increasing or decreasing their participation in the search. A small minimum probability is considered for the heuristics so they are not completely removed from the search, as they can present bad results at some stage, but could perform well ahead in the search procedure.

3.8 Benchmark problems

Benchmark problems are often used in the literature to assess the performance of a multi-objective optimizer or to compare two or more algorithms. This type of problem presents advantages over using real-world problems, since their real Pareto front is often known, their difficulty degree can be controlled, and in most problems, the number of objectives and decision variables can also be controlled.

The multi-objective search community has been using several benchmark problems over the years, among them, we can highlight the ZDT (Zitzler et al., 2000) family of benchmarking problems. Although being widely used to test multi-objective optimizers, the problems of this family are not scalable regarding the number of objectives, preventing its use with more than two objective functions.

A family of benchmarking problems that is attracting the attention of the community is presented in the WFG (Huband et al., 2006) toolkit. WFG allows researchers to define scalable, multi-objective test problems with Pareto fronts of different geometries, parameter dependencies, modality, bias and fitness landscapes. Also, using WFG the authors proposed a set of nine test functions covering a wide range of problem attributes (von Lücken et al., 2014). In all the nine WFG problems, the objective values are to be *minimized*. The number of objectives m can be set to any value greater than one. The number of decision variables is $n = d + p$, where d is the number of distance related variables and p is the number of position related ones.

Another widely used (von Lücken et al., 2014) family of benchmark problems is the DTLZ problem suite (Deb et al., 2002). It was the first set of functions specifically designed to study scalability issues of multi-objective algorithms. DTLZ is composed of seven problems named as DTLZ1 to DTLZ7, which are scalable in both decision and objective dimensions. Each of these problems represents different characteristics (spherical, linear, discontinuous and degenerate) of the Pareto-optimal front. It is worth noting that according to Huband et al. (2006), despite DTLZ5 and DTLZ6 being both claimed to be problems with degenerate Pareto optimal fronts, this is untrue for instances with four or more objectives (von Lücken et al., 2014). In all the seven DTLZ problems, the objective values are to be *minimized*. The number of objectives m

can be set to any value greater than one. The number of decision variables is $n = m + p - 1$, where the number of distance variables p also can be set to any value, however, the number of position variables is fixed in $m - 1$. By increasing or decreasing the value of p, the problem can be made easier or harder.

A potential drawback of these benchmarks functions (except for DTLZ7 and WFG8) is that the decision variables are split into two main groups according to their relationship with the fitness landscape: distance variables and position variables. *Distance variables* are related to the convergence characteristic of sets of solutions for the problem. By changing a single distance variable of a solution, we generate a new solution that dominates, is equal to or is dominated in relation to the previous. These solutions will never be strictly non-dominated in relation to each other. *Position variables* are related to the spread of solutions, and by modifying an individual position variable of a solution, we only generate a new solution that is incomparable (non-dominated) or equal to the original solution (Huband et al., 2006). Although this division allows to separately evaluate the behavior of optimizers in terms of spread and convergence, in real-world problems a more refined classification of variables is often required in order to model the characteristics of the problems (Brownlee and Wright, 2012).

A summary of the characteristics of the DTLZ and WFG problems is presented in Table 3.1, adapted from Li et al. (2015).

Table 3.1: Characteristics of the functions included in the DTLZ and WFG benchmarks

Problem	Pareto front geometry	Challenges of decision space
DTLZ1	linear	multi-modal
DTLZ2	concave	none
DTLZ3	concave	multi-modal
DTLZ4	concave	biased
DTLZ5	concave, degenerate	none
DTLZ6	concave, degenerate	multi-modal
DTLZ7	mixed, disconnected	multi-modal
WFG1	mixed	biased
WFG2	convex, disconnected	multi-modal, non-separable
WFG3	linear, degenerate	non-separable
WFG4	concave	multi-modal
WFG5	concave	deceptive
WFG6	concave	non-separable
WFG7	concave	biased
WFG8	concave	biased, non-separable
WFG9	concave	biased, multi-modal, deceptive, non-separable

The geometry of a Pareto front can be characterized in concave, convex, linear, mixed or degenerate. *Convex* fronts are curved outwards toward better solutions. *Concave* fronts are curved inwards, away from better solutions. A *Linear* front is both concave and convex at the same time. *Mixed* fronts have two or more connected subsets that are each convex, concave or linear, but not all of the same type. A *Degenerate* front is a front that is of lower dimension than de objective space in which it is embedded, less one. For example, a front that is a line segment in a three objective problem is degenerate. A Pareto front can also be *connected* or disconnected, that is usually referred to as *discontinuous* (Huband et al., 2006; Luke, 2013).

There are some properties that the objective functions of optimization problems can present that are challenging to optimizers. The most usual are (Huband et al., 2006):

- *Multi-modality*: an objective function is multi-modal when it has multiple local optima. An objective function with only a single optimum is *unimodal*.

- *Deceptive* functions have a special kind of multi-modality, it has to have at least two optima, a true optimum and a deceptive optimum, but the majority of the search space must favor the deceptive optimum.

- *Bias* means that there is a significant variation in distribution between vectors sampled in the decision space and their mapping in the objective space.
- *Non-separability*: an objective function is non-separable, if we cannot optimize it by considering each decision variable in turn, independently of one another. An objective function whose decision variables can be optimized separately is *separable*.

3.9 Performance metrics

Assessing the performance of a multi-objective optimizer is not a trivial task, however, some characteristics are desired in a Pareto front/set generated by an optimization algorithm. The three main requirements for a multi-objective optimizer are (Adra, 2007):

- *Convergence:* The PF_k achieved for a MOP is required to be as close as possible to the PF_t.
- *Diversity:* Because of the non-existence of an ideal single solution in multi-objective optimization problems with competing objectives, and due to the fact that the global trade-off surface can potentially present an infinite number of solutions, PF_k is also required to be well spread and uniformly cover wide areas of PF_t. Diversity of solutions is conventionally preferred in the objective space to present to the decision-maker a well distributed set of solutions to choose from, based on certain preferences such as objective priorities or region of interest. Diversity of solutions is however not restricted to the objective space, and can be a desired requirement in the decision space of some applications (Preuss et al., 2010).
- *Pertinence:* As the dimensionality of the problem increases, the visualization of the optimization process becomes a problem. The decision-maker is usually interested in sub-regions of the search space that turn the decision-making and the optimization processes more practical and efficient. Therefore, convergence and diversity of solutions are particularly required in the pertinent areas of the space, or regions of interest.

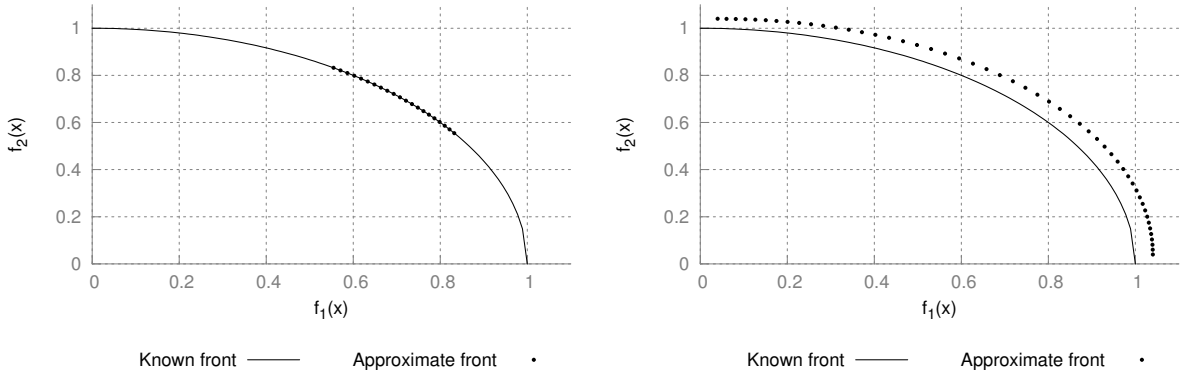
Since in this work we do not use regions of interest, the pertinence characteristic is not taken in consideration and the assessment basis in our work will be the convergence and diversity. In this scenario, a good optimizer must provide a set of solutions that has good convergence and diversity. However, these two goals are usually conflicting, hence the main challenge of an optimizer is to enhance both simultaneously.

Figure 3.13 shows examples of Pareto fronts for a bi-objective minimization problem, the gray line represents the true front and the black points represent non-dominated solutions contained in an approximate Pareto front. The approximation of Figure 3.13(a) presents optimal convergence, since their points belong to the true front, however, they do not present good diversity as the solutions only cover a small portion of the true Pareto front.

Figure 3.13(b) on the other hand, presents a diverse Pareto front, since the solutions cover almost entirely the true front, however in this case the convergence is not good, since the points are far from the true front.

Plotting the fronts achieved by an optimizer in problems presenting two or three objectives can be useful for comparing their performance, however there are cases where those fronts are visually very similar. There is also the case of problems involving more than three objectives, where directly plotting the objective values is not possible.

The performance of optimizers in such cases can be assessed and compared by using several quality indicators. Following we present some of them.



(a) Well converged front

(b) Well distributed front

Figure 3.13: Examples of approximated fronts

3.9.1 GD_p and IGD_p

Generational Distance (GD) (Veldhuizen and Lamont, 1998) is an indicator that reports how far, on average, PF_k is from PF_t. This is a Pareto non-compliant metric and requires PF_t to be known (Coello et al., 2006). However in some works a combination of all the non-dominated solutions from all runs of all algorithms is used as PF_t for GD calculation (Jaimes et al., 2013). GD is defined by Equation (3.31).

$$GD(PF_k, PF_t) \triangleq \frac{(\sum_{i=1}^{N} d_i^p)^{1/p}}{N} \tag{3.31}$$

where N is the number of vectors in PF_k, $p = 2$ and d_i is de Euclidean distance between each objective vector and the nearest member of PF_t. A result of 0 indicates $PF_k = PF_t$; any other value indicates PF_k deviates from PF_t (Veldhuizen and Lamont, 1998).

Inverted Generational Distance (IGD) (Coello and Cortés, 2005), measures the smallest distance between each point in PF_t and the points in PF_k. IGD allows observing if PF_k is well diversified and close to PF_t. It is calculated by interchanging the roles of PF_k and PF_t in the GD definition:

$$GD(PF_k, PF_t) = IGD(PF_t, PF_k) \tag{3.32}$$

It is important to conduct a joint analysis of GD and IGD since the GD measures only the convergence. Hence, if the solutions are clustered in a small region of the objective space, but close to PF_t, the GD value of this set will be low, which does not necessarily mean it is a good approximation.

The GD indicator is used in many studies, but it is not accepted by all researchers. The main reason for this, identified by Schutze et al. (2012), is its averaging strategy, demonstrated by the following example: assume a given point $\boldsymbol{u}$ where its distance to PF_t is 1. Now, define PF_k as a set given by N copies of $\boldsymbol{u}$, i.e., $PF_k = \{\boldsymbol{u}, ..., \boldsymbol{u}\}$. Then, for the "averaged" distance of PF_k to PF_t it holds (Schutze et al., 2012)

$$GD(PF_k, PF_t) = \frac{(\sum_{i=1}^{N} 1^p)^{1/p}}{N} = \frac{\sqrt[p]{N}}{N} \tag{3.33}$$

We see that: 1) with increasing the number N, the approximation quality gets "better", but the approximation has apparently not changed; 2) by increasing the size of PF_k, it converges even to a "perfect" approximation, i.e.

$$\lim_{N\to\infty} GD(PF_k, PF_t) = 0 \tag{3.34}$$

This result in (3.34) can be generalized: instead of copies, it can be considered small perturbations of $\boldsymbol{u}$. Or, if $\boldsymbol{u}$ is bounded, even any solution inserted in PF_k with $|PF_k| = N$ can be chosen, regardless if the entries $\boldsymbol{u}$ of PF_k are dominated or not, or how far $\boldsymbol{u}$ is away from the Pareto front. Hence, in the context of an optimizer, it is advantageous to fill the archive with further solutions, even dominated ones, since typically larger sets yield better GD values. In the community, it has been established to fix the repository and population sizes in order to allow a comparison of different algorithms. However, this leads to trouble for algorithms based on archivers that are not bounded by a predefined value. A "perfect" archiver in this case is the one that accepts all candidate solutions. An effect which is certainly not desired (Schutze et al., 2012).

A slightly modified version of GD, called GD_p is proposed by Schutze et al. (2012) to avoid this undesired aspect, improving significantly the Pareto compliance of GD. In GD_p the power mean is used to average the distances between the elements, becoming:

$$GD(PF_k, PF_t) \triangleq \left(\frac{\sum_{i=1}^{N} d_i^p}{N} \right)^{1/p} \tag{3.35}$$

GD_p does not have the unwanted characteristic previously discussed, hence large sets do not necessarily achieve good indicator results any more, in the above example, now we have $GD_p(PF_k, PF_t) = 1$ for all numbers $N \in \mathbb{N}$ (Schutze et al., 2012).

The same modification of GD to GD_p is proposed for IGD to IGD_p. In multi-objective optimization, a suitable discretization of the Pareto front has to be chosen. Analog to the discussion for GD, the IGD value gets better when choosing a finer discretization of the Pareto front. This problem can be avoided by fixing a discretization of the Pareto front, but this is also an unwanted effect (Schutze et al., 2012).

3.9.2 Hypervolume

Another popular metric for comparing the performances of multi-objective optimizers is hypervolume (Zitzler and Thiele, 1999). The hypervolume of a set of solutions measures the size of the portion of the objective space that is dominated by those solutions. Hypervolume captures in one scalar both, the closeness of the solutions to the optimal set and their spread across objective space. It also has nicer mathematical properties than other metrics, being the only type of indicator that is strictly monotonic[4]. However hypervolume is very sensitive to the scaling of the objectives and to the presence of extremal points (Bader et al., 2010; Coello et al., 2006; Phan and Suzuki, 2013; While et al., 2012).

[4]When a Pareto set approximation dominates another, the indicator value of the former will be greater than the latter.

The hypervolume of a set relative to a reference point is measured, usually the nadir (anti-optimal or "worst possible") point in space. The greater the hypervolume value of a set is, the better that set is taken to be. The hypervolume is defined as follows (Bringmann et al., 2013)

$$HV(PF_k) := VOL\left(\bigcup_{(u_1,\ldots,u_m)\in PF_k} [r_1, u_1] \times \ldots \times [r_m, u_m]\right) \tag{3.36}$$

where $VOL(.)$ is the usual Lebesgue measure and the reference point $\boldsymbol{r}$ is the nadir (anti-optimal or "worst possible") point in space.

The main disadvantage of the hypervolume indicator is that its calculation is computationally expensive, even the best known algorithms for computing the hypervolume have running times exponential in the number of objectives, which restricts the use of hypervolume-based methods to problems with no more than ten objectives (Bader et al., 2010; Phan and Suzuki, 2013).

In this work, up to eight objectives we employ the exact hypervolume calculation, however, when considering more than eight objectives, an approximated version, based on Monte Carlo sampling (Bader et al., 2010) is used.

3.9.3 *R*2

Another recommended quality indicator is *R*2 (Hansen and Jaszkiewicz, 1998). It was originally proposed to assess the relative quality of two approximation sets. It is an indicator that simultaneously evaluates the convergence and diversity of a Pareto front approximation and presents a low computational cost (Brockhoff et al., 2012).

Assuming the standard weighted Tchebytcheff function with a particular reference point $\boldsymbol{z}^*$, the indicator can be used to assess the quality of a single Pareto front PF_k against $\boldsymbol{z}^*$ (Brockhoff et al., 2012; Phan and Suzuki, 2013). The weighted Tchebytcheff function depends on a set of weight vectors $\boldsymbol{w} = (w_1, ..., w_m) \in \boldsymbol{W}$. These weights are usually chosen uniformly distributed over the weight space. In this case, the *R*2 indicator can be described as:

$$R2(PF_k, \boldsymbol{W}, \boldsymbol{z}^*) = \frac{1}{|\boldsymbol{W}|} \sum_{\boldsymbol{w}\in\boldsymbol{W}} \min_{\boldsymbol{u}\in PF_k} \{ \max_{j\in\{1,\ldots,m\}} \{w_j | z_j^* - u_j|\}\} \tag{3.37}$$

An ideal point is usually used as the reference point $\boldsymbol{z}^*$. An ideal point is a point that is never dominated by any feasible solution in the objective space, i.e., (0,0) for a bi-dimensional objective space where all the objective values are greater than or equal to 0. The set of weight vectors is generated using the approach proposed by Das and Dennis (1998), that places points on a normalized hyper-plane which is equally inclined to all objective axes and has an intercept of one on each axis (an $m-1$ dimensional unit simplex).

A lower *R*2 value indicates that an individual set is closer to the reference point. $R2 = 0$ when an individual is positioned on the reference point (Phan and Suzuki, 2013).

Hypervolume so far is the only known indicator which fulfills the property of strict monotonicity as commented before. *R*2 is only weakly monotonic[5]. In contrast, the *R*2 indicator is often preferred over the hypervolume for two reasons. One of them is the lower computational cost, the other is the distributions obtained using the hypervolume, that are biased toward the knee

[5]When a Pareto set approximation dominates another, the indicator value of the former will be smaller than or equal to the latter.

regions[6] of the Pareto front. The $R2$ is assumed to result in a more uniform distribution (Brockhoff et al., 2012).

3.10 Statistical tests

In multi-objective research community, it is very usual to compare optimizers, say A and B, in order to determine if they outperform one another, or if their performance is similar. However, since most of the optimizers are stochastic, this comparison cannot be made based on the indicator results for only one run of the algorithms, since the differences can be attributed to the different random number generator. To eliminate the possibility of random interference in the results, the algorithms can be run n times, and n needs to be large (Luke, 2013).

This could be made by running A and B a billion times and comparing their means, however it could take some time. To state with confidence that one algorithm is better than another with some smaller number of runs, a hypothesis test (Luke, 2013) can be used.

In this test, two hypothesis are used, the null hypothesis and the alternative hypothesis. The null hypothesis claims that there is no difference between the algorithms, that is, the perceived difference is just due to the random numbers. The alternative hypothesis represents the presence of an effect or a difference (i.e., A and B are significantly different). When applying a statistical procedure to reject any of these hypotheses, a level of significance is used to determine at which level the hypothesis may be rejected (Derrac et al., 2011; Luke, 2013).

Instead of determining a priori a level of significance, it is possible to compute the smallest level of significance that results in the rejection of the null hypothesis. This is the definition of the p-value, which is the probability of obtaining a result at least as extreme as the one that was actually observed, assuming that the null hypothesis is true. The smaller the p-value, the stronger is the evidence against the null-hypothesis (Derrac et al., 2011).

A hypothesis test estimates this significance. There are several different hypothesis tests in the literature that can be classified as parametric and non-parametric. Parametric tests rely solely on the mean, variance and sample count of the results. This is because they make a huge assumption: that the results produced by A and B are drawn from a normal (Gaussian) distribution. In meta-heuristics scenarios that is almost never true (Luke, 2013).

Another approach is the use of non-parametric hypothesis tests. Besides their original definition for dealing with nominal or ordinal data, they can also be applied to continuous data by conducting ranking-based transformations, adjusting the input data to the test requirements. As a result, such tests are much less sensitive, but they are not fooled by assumptions about the distribution of the results. If an algorithm passes a non-parametric test, few can criticize it (Derrac et al., 2011; Luke, 2013).

Non-parametric tests can perform two classes of analysis: pairwise comparisons and multiple comparisons. Pairwise comparisons are the simplest kind of statistical tests that a researcher can apply within the framework of an experimental study. Such tests are used to compare the performance of two algorithms when applied to a common set of problems. In multi-problem analysis, a value for each pair algorithm/problem is required (often an average value from several runs) (Derrac et al., 2011).

A simple, yet safe and robust non-parametric test for pairwise statistical comparison is the Wilcoxon signed ranks test (Wilcoxon, 1945). This test is used for answering the following question: do two samples represent two different populations? It is a non-parametric procedure

[6]Knees are solutions where a small improvement in one objective leads to a large deterioration in at least one other objective (Coello et al., 2009).

employed in hypothesis testing situations involving a design with two samples. This is analogous to the paired t-test in non-parametric statistical procedures; therefore it is a pairwise test that aims to detect significant differences between two sample medians, i.e., the behavior of two algorithms (Derrac et al., 2011).

Despite the good characteristics of the Wilcoxon test, one of the most frequent situations where the use of statistical procedures is requested is in the joint analysis of the results achieved by various algorithms. The groups of differences among these methods (also called blocks) are usually associated with the problems met in the experimental study. For example, in a multiple problem comparison, each block corresponds to the results offered over a specific problem. When referring to multiple comparison tests, a block is composed of three or more subjects or results, each one corresponding to the performance evaluation of the algorithm over the problem (Derrac et al., 2011).

In pairwise analysis, if we try to extract a conclusion involving more than one pairwise comparison, we will obtain an accumulated error coming from its combination. In statistical terms, we are losing the control of the Family-Wise Error Rate (FWER), defined as the probability of making one or more false discoveries among all the hypotheses when performing multiple pairwise tests. Therefore pairwise tests, such as Wilcoxon test should not be used to conduct various comparisons involving a set of algorithms, because the FWER is not controlled (Derrac et al., 2011).

A widely used statistical test for multiple comparisons is the Friedman test (Friedman, 1937). The Friedman test is a non-parametric statistical procedure for comparing more than two samples that are related. The parametric equivalent to this test is the repeated measures analysis of variance (ANOVA) (Corder and Foreman, 2014).

Another well-known statistical test for multiple comparisons is the Kruskal-Wallis test (Kruskal and Wallis, 1952). The Kruskal-Wallis test is a non-parametric statistical procedure for comparing more than two samples that are independent or not related. The parametric equivalent to this test is the one-way analysis of variance (ANOVA) (Corder and Foreman, 2014).

In both, Friedman and Kruskal-Wallis tests, when the test leads to significant results, then at least one of the samples is different from the others. However, the test does not identify where the differences occur. Moreover, it does not identify how many differences occur. In order to identify the particular differences between sample pairs, a researcher might use samples contrasts, or post-hoc tests (Corder and Foreman, 2014).

In this work, the results measured with the quality indicators in 30 independent runs of the algorithms on each problem instance are submitted to the Kruskal-Wallis test at 5% significance level. When there are many results being compared and it is hard to take general conclusions, we summarize the results by averaging the 30 runs of each algorithm on each problem instance (problem/objective number). By averaging these results and considering them together, they become related, or dependent. Hence we employ a Friedman test, also at 5% significance level on this summarized data set. When significant differences were found in any of the tests, we conduct a post-hoc analysis using the Nemenyi (Nemenyi, 1963) test to identify particular differences between samples (Demsar, 2006).

The results of the statistical tests are presented as tables showing the indicator values achieved for the compared algorithms in the form of the ranks used by the applied statistical test. The number in parentheses indicates the final classification of the algorithms, where smaller rank values are better. The algorithm with the best rank is highlighted in bold font.

When calculating the final ranks, in case of statistical tie (algorithms presenting no statistically significant difference), the final rank of each of the tied algorithms is equal to the average of the ranks that would be assigned to them.

3.11 Discussion

This chapter presented the theoretical foundation that will be used throughout this work. Basic concepts and notation of single and multi-objective optimization problems, PSO and MOPSO, including different strategies for leader selection and archiving. The decomposition-based approaches MOEA/D and MOEA/DD were also introduced, as well as two different EDAs and three hyper-heuristic approaches. The benchmark problems, performance indicators and statistical tests employed to validate and compare different algorithms were also described.

Chapter 4

Using hyper-heuristics to select leader and archiving methods for MOPSO

Previous works have pointed that selecting a proper combination of leader and archiving methods, which is a challenging task because they are problem-dependent, helps a MOPSO algorithm to improve its performance (Britto and Pozo, 2012; Castro Jr. et al., 2012).

An option to deal with this challenge is using hyper-heuristics. Hyper-heuristics can be used to dynamically select, from a set of low-level heuristics, the best components for an optimization algorithm to effectively solve a given problem. The use of hyper-heuristics to dynamically choose the components of MOPSO presents two significant advantages. First, it eliminates the need to manually select the components, and second, it allows the application of different leader and archiving methods during the search. Alternating these components allows combining methods that prioritize convergence and others that prioritize diversity, reducing the probability of being stuck in local optimal fronts.

In this chapter, we propose a new MOPSO framework called Hyper-heuristic Multi-Objective Particle Swarm Optimization (H-MOPSO). This framework can use any selection hyper-heuristic to dynamically select a combination of leader and archiving methods during the search. To investigate the behavior of H-MOPSO under different scenarios, as well as to compare it to state-of-the-art algorithms, a number of experimental studies were conducted:

In the first study, our goal is to assess if the H-MOPSO framework is able to guide the search through the use of hyper-heuristic. To achieve this goal, we compared the H-MOPSO using the simple roulette wheel hyper-heuristic to all its low-level heuristics used separately.

In a second experimental study, we compare the performance of H-MOPSO using four heuristic selection methods, two state-of-the-art from the literature: Adaptive Choice Function (ACF) (Drake et al., 2012; Gonçalves et al., 2015b) and Fitness-Rate-Rank-based Multi-Armed Bandit (FRRMAB) (Gonçalves et al., 2015a; Li et al., 2014). The remaining two heuristic selection methods are the simple roulette wheel previously used and a random hyper-heuristic that employs no learning.

In a third study, two characteristics of the hyper-heuristic are configured: the set of low-level heuristics to be dynamically replaced (i.e., only archiving, only leader selection or both methods simultaneously), and the interval used to replace the low-level heuristics. Our goal here is to identify if changing only a subset of low-level heuristics is best than changing all. It means if selecting an appropriate option for one method (archiving or leader selection) is better than selecting the proper options for both together. Moreover, we want to calibrate the interval used to replace the low-level heuristics to give them more time (in iterations number) to influence the search, hence allowing to evaluate if subsequent applications of the same hyper-heuristic can intensify its performance gain.

At a final study, in order to validate the performance of H-MOPSO, we compare it to the state-of-the-art MOEA/D-FRRMAB (Li et al., 2014) hyper-heuristic framework. For this experiment, we used the best parameter values and heuristic selection method found in the previous experiments.

This Chapter is organized as follows: first the H-MOPSO framework is described in Section 4.1, then Section 4.2 presents the aforementioned experimental studies. A discussion of the content presented in this chapter is available in Section 4.3.

4.1 Hyper-MOPSO

To expand a PSO algorithm into a MOPSO, usually two main modifications are made: the creation of an external archive, or repository, to store the best non-dominated solutions found so far, and the use of a leader selection method to select a global leader for each particle among a set of equally good solutions according to some criterion. As seen in previous works (Britto and

Pozo, 2012; Castro Jr. et al., 2012), leader and archiving methods have a significant impact in the optimization process of a MOPSO, however these methods are more or less suitable to different problems and there is no single method that excels in all problems.

An option to deal with this challenge of selecting appropriate components of optimization algorithms, is the use of hyper-heuristics. Hyper-heuristics can be used to dynamically select, from a set of low-level heuristics, the best components for an optimization algorithm to effectively solve a given problem.

Here, we consider as low-level heuristics a combination of archiving and leader selection methods. The archiving and leader selection methods used here are those presented in Sections 3.3.1 and 3.3.2 respectively, and the low-level heuristics generated with their combinations are presented in Table 4.1.

Table 4.1: Low-level heuristics available for the heuristic selection methods

Heuristic	Leader selection	Archiver
$H1$	CD	CD
$H2$	CD	Ideal
$H3$	CD	MGA
$H4$	Sigma	CD
$H5$	Sigma	Ideal
$H6$	Sigma	MGA
$H7$	NWSum	CD
$H8$	NWSum	Ideal
$H9$	NWSum	MGA

Following, we propose a framework called Hyper-heuristic Multi-Objective Particle Swarm Optimization (H-MOPSO). H-MOPSO dynamically selects the best archiving and leader selection methods for a MOPSO. Further, the framework allows the implementation of different heuristic selection and move acceptance methods.

The H-MOPSO framework is illustrated in Algorithm 9. The highlighted steps are the hyper-heuristics steps introduced in a generic MOPSO. In the first step, the algorithm initializes the swarm. Following, the repository is initialized with the non-dominated solutions from the swarm, and the algorithm initializes the hyper-heuristic strategy.

After initialization steps, the algorithm enters the main loop (lines 4 to 16). The first step of the main loop is to select a low-level heuristic, using some heuristic selection method (line 5). Next, for each particle, the MOPSO steps are executed (lines 6 to 12). The currently selected leader method is applied. The particle velocity and position are updated. A turbulence method is used (mutation). The fitness of the particle is evaluated, and the local best leader is updated.

Next, the repository is updated, using the currently selected archiving method (line 13). The performance of the applied low-level heuristic is evaluated in line 14, and the probabilities of each low-level heuristic being selected are updated according to the hyper-heuristic mechanism. In line 15, the move acceptance method is used to decide if the new repository is going to be accepted (replace the old one) or discarded (keep using the old one). When the stop criterion is reached, the algorithm output is the solution set contained in the repository (line 17).

The first step introduced in this generic MOPSO is the initialization of the hyper-heuristic. This step is specific for each hyper-heuristic, and is responsible for initializing the data structures with default values.

The second step is the heuristic selection itself. Any heuristic selection can be used, however we have selected the ones presented in Section 3.7 with the addition of a random heuristic selection as control method.

The third step is an evaluation of the performance of the low-level heuristic selected. In this step we evaluate the new and old repositories using the $R2$ indicator (see Section 3.9.3), and the

Algorithm 9: H-MOPSO

```
1  Initialize swarm;
2  Initialize repository;
3  Initialize hyper-heuristic;
4  while not reached the stop criteria do
5      Heuristic selection;
6      foreach particle ∈ swarm do
7          Select leader;
8          Update Position;
9          Turbulence;
10         Evaluation;
11         Update Personal Best;
12     end
13     Update repository (swarm);
14     Evaluate heuristic performance;
15     Move acceptance;
16 end
17 return solutions in repository;
```

quality information used by the selection methods for learning, is defined as the normalized quality difference between the new repository ($R2_{new}$) and the old ($R2_{old}$) as follows (Equation 4.1):

$$Quality = \frac{R2_{old} - R2_{new}}{R2_{old}} \tag{4.1}$$

The last step is the move acceptance criterion, where we decide if the new repository will replace the old one. In this work we have used the improving and equal (IE) move acceptance method due to its simplicity and good results obtained by Bilgin et al. (2007). In IE a new solution is accepted if it improves or maintains the previous score value.

An important remark about the archiving methods is that until the repository size limit exceeds, every archiving method will have the same behavior, i.e., to accept all non-dominated solutions. This means that, at the initial iterations, the quality of different archiving methods cannot be estimated. Therefore, it makes sense to begin the Hyper-heuristic selection method only when the repository becomes full. There is the possibility of selecting only the leader method until the repository is full, however it would add complexity to the code. Another issue about H-MOPSO refers to the decision about which MOPSO to use and. Here the Speed-constrained Multi-objective PSO (SMPSO) was selected because of its good results described in the literature (Durillo et al., 2009; Nebro et al., 2009).

4.2 Empirical study

This section presents the empirical studies conducted to investigate the behavior of H-MOPSO under different scenarios. Section 4.2.1 presents the general parameters used in the experiments, additional specific parameters are available at the beginning of each section. The first experimental study that investigates the ability of H-MOPSO of guiding the search is presented in Section 4.2.2. The third experimental study is conducted to identify which is

the best heuristic selection method to be used within H-MOPSO, hence the performance of H-MOPSO using four heuristic selection methods (ACF, FRRMAB, Roulette and random) is compared in Section 4.2.3. Section 4.2.4 shows the calibration of two important parameters of H-MOPSO: the set of low-level heuristics to be dynamically replaced (only archiving, only leader, or both simultaneously) and the interval used to replace the low-level heuristics. A final study, conducted to validate the performance of H-MOPSO by comparing it to the state-of-the-art MOEA/D-FRRMAB (Li et al., 2014) is available in Section 4.2.5.

4.2.1 Experimental setup

The basic parameters used in this study for the H-MOPSO follow the same proposed for SMPSO (Nebro et al., 2009) since H-MOPSO is based on it. The parameters C_1 and C_2 varies randomly in [1.5,2.5]. The inertia ω is 0.1 and the polynomial mutation (turbulence) (Deb, 2009) is applied to 15% of the population.

For the Roulette heuristic selection, the initial probabilities are the same for all the low-level heuristics. The minimum probability of each heuristic was set to 0.5% to prevent it from being removed from the search. Moreover, the increment (or decrement) in the probability of the selected heuristic is set to one tenth of the initial probabilities[1].

For the ACF heuristic selection, the considered parameters were: Scaling Factor (SF) of 0.1, f_1 and f_2 used the accumulated values as in the original algorithm. For the FRRMAB heuristic selection, the used parameters were: scaling factor (C) set to 10, decaying factor for the reward (D) set to 1 and sliding window size (W^w) set to 10. These parameters were empirically calibrated before the experiments, and the best values were used. The move acceptance criterion utilized in all variants was Improving and Equal (IE).

The nine low-level heuristics, composed of a leader selection and an archiving method, and available to be selected by each heuristic selection strategy are summarized in Table 4.1.

The remaining parameters, like population and repository sizes, stopping criterion and parameters specific to the compared algorithms are presented in each section, since they can be different according to the experimental study conducted. Since each of the studies investigate a different scenario, the benchmark problems, computational budget and quality indicators were set in the best way to investigate each scenario. For instance, in the next experiment, our goal is not to measure the performance of the algorithm, but to assess if a simple hyper-heuristic is able to guide the search, hence we employ the $R2$ indicator, like the one used in the internal evaluation of the algorithm.

4.2.2 H-MOPSO vs. low level heuristics

In this section, we followed the parameters used by Nebro et al. (2009), and used as stopping criterion the number of iterations, set to 100. The number of particles and the size of the repository were also set to 100. The entire DTLZ (Deb et al., 2002) family of problems was used (DTLZ1 - DTLZ7) with number of objectives ranging from 2 to 20. Since we are interested only in assessing the ability of H-MOPSO in guiding the search, we use the same indicator used in its inner work, the $R2$.

Tables 4.2 and 4.3 show the Kruskal-Wallis ranks of the $R2$ results achieved by the algorithms. For better visualization, the average of the 30 runs of each algorithm is shown through Figures 4.1, 4.2, 4.3 and 4.4. In these figures, each line corresponds to the average indicator result of an algorithm for each objective number.

[1]This parameter was calibrated during the development of the algorithm

Table 4.2: Kruskal-Wallis ranks for the $R2$ indicator

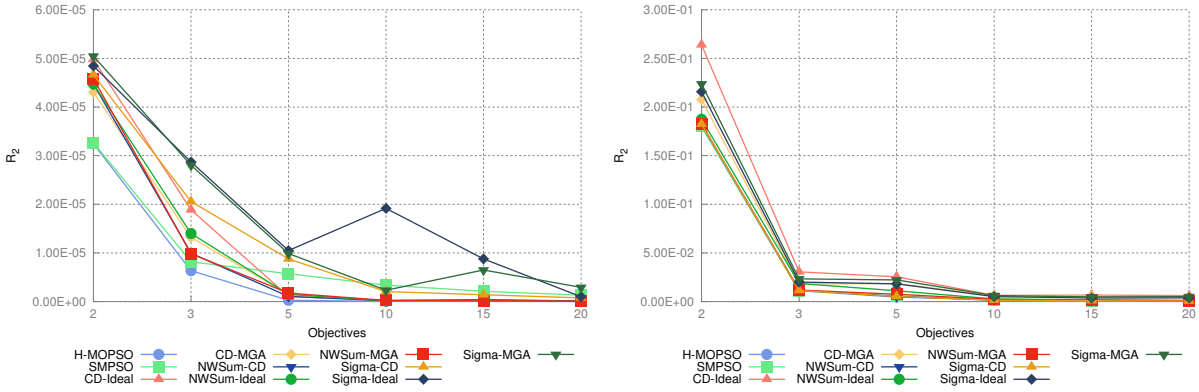
Obj.	Algorithms	DTLZ1	DTLZ2	DTLZ3	DTLZ4	DTLZ5	DTLZ6	DTLZ7
2	H-MOPSO	**17.70 (1.50)**	**30.23 (1.50)**	**20.70 (1.00)**	44.70 (2.50)	**17.43 (1.50)**	**34.23 (1.50)**	**16.93 (2.00)**
	SMPSO	**44.10 (1.50)**	**30.77 (1.50)**	107.70 (4.00)	**16.30 (1.50)**	**43.57 (1.50)**	**54.37 (1.50)**	**44.40 (2.00)**
	CD-Ideal	216.90 (7.00)	267.97 (9.50)	202.00 (8.00)	256.07 (8.00)	263.00 (9.50)	192.00 (6.50)	263.00 (8.50)
	CD-MGA	141.67 (6.00)	208.50 (6.50)	171.03 (6.00)	200.63 (8.00)	221.53 (8.00)	129.07 (6.50)	210.93 (7.50)
	NWSum-CD	161.10 (6.50)	136.97 (6.00)	118.20 (4.00)	105.07 (3.50)	133.73 (5.50)	169.13 (6.50)	89.83 (2.50)
	NWSum-Ideal	159.63 (6.50)	178.37 (6.00)	112.70 (4.00)	119.87 (4.00)	171.40 (6.00)	171.47 (6.50)	177.43 (6.50)
	NWSum-MGA	174.67 (6.50)	142.00 (6.00)	98.73 (4.00)	115.40 (3.50)	140.83 (5.50)	173.17 (6.50)	121.67 (4.50)
	Sigma-CD	185.30 (6.50)	136.47 (6.00)	226.10 (8.00)	212.37 (8.00)	140.03 (5.50)	191.07 (6.50)	201.00 (7.50)
	Sigma-Ideal	193.07 (6.50)	187.23 (6.00)	214.90 (8.00)	217.63 (8.00)	187.83 (6.00)	196.67 (6.50)	173.43 (6.50)
	Sigma-MGA	210.87 (6.50)	186.50 (6.00)	232.93 (8.00)	216.97 (8.00)	185.63 (6.00)	193.83 (6.50)	206.37 (7.50)
3	H-MOPSO	**15.60 (1.50)**	**15.63 (2.00)**	**16.80 (1.00)**	**21.27 (2.00)**	**25.77 (1.50)**	**32.33 (1.50)**	**20.40 (1.00)**
	SMPSO	68.53 (3.00)	63.03 (3.00)	126.13 (4.50)	55.20 (2.50)	**35.23 (1.50)**	79.73 (2.50)	108.93 (4.50)
	CD-Ideal	188.77 (7.50)	265.27 (8.00)	176.20 (5.50)	252.00 (8.00)	270.10 (9.50)	174.73 (6.50)	164.83 (5.50)
	CD-MGA	159.97 (5.50)	218.60 (8.00)	144.60 (4.50)	224.40 (7.50)	215.47 (7.00)	113.10 (4.50)	169.17 (5.50)
	NWSum-CD	115.07 (4.00)	115.93 (3.50)	117.83 (4.50)	82.00 (2.50)	148.10 (6.00)	201.43 (7.00)	246.90 (9.00)
	NWSum-Ideal	136.13 (5.00)	198.10 (8.00)	111.07 (4.50)	178.63 (6.50)	178.10 (6.00)	212.40 (7.00)	221.60 (8.00)
	NWSum-MGA	115.03 (4.00)	121.07 (3.50)	103.37 (4.50)	112.90 (3.50)	171.27 (6.00)	201.63 (7.00)	228.53 (8.00)
	Sigma-CD	193.33 (7.50)	68.33 (3.00)	222.10 (8.50)	189.70 (7.50)	111.57 (5.50)	146.10 (6.00)	125.43 (4.50)
	Sigma-Ideal	257.40 (8.50)	213.83 (8.00)	253.17 (9.00)	192.17 (7.50)	179.43 (6.00)	170.97 (6.50)	112.73 (4.50)
	Sigma-MGA	255.17 (8.50)	225.20 (8.00)	233.73 (8.50)	196.73 (7.50)	169.97 (6.00)	172.57 (6.50)	106.47 (4.50)
5	H-MOPSO	**16.73 (1.00)**	**18.67 (2.00)**	**23.13 (1.00)**	**18.83 (2.00)**	**38.93 (2.50)**	**28.43 (2.00)**	**20.83 (2.00)**
	SMPSO	160.27 (5.00)	103.17 (3.50)	130.00 (4.50)	**53.97 (2.00)**	109.73 (4.50)	99.10 (3.00)	220.77 (8.00)
	CD-Ideal	104.00 (4.50)	256.40 (8.50)	116.60 (4.50)	193.90 (6.50)	284.23 (9.50)	107.27 (4.00)	143.13 (4.50)
	CD-MGA	119.13 (4.50)	234.60 (8.00)	120.80 (4.50)	171.17 (6.50)	165.07 (6.00)	69.97 (2.50)	154.70 (6.00)
	NWSum-CD	121.67 (4.50)	69.63 (3.00)	131.90 (4.50)	**83.10 (2.00)**	178.77 (6.00)	203.33 (7.50)	251.67 (8.50)
	NWSum-Ideal	114.60 (4.50)	177.80 (7.00)	105.57 (4.50)	267.83 (10.00)	236.00 (8.00)	249.60 (8.50)	210.67 (7.50)
	NWSum-MGA	137.47 (4.50)	130.27 (4.00)	135.47 (4.50)	189.10 (6.50)	134.03 (5.00)	183.57 (7.50)	238.73 (8.50)
	Sigma-CD	231.87 (8.50)	61.20 (3.00)	244.93 (9.00)	173.87 (6.50)	84.67 (3.00)	161.17 (6.00)	116.07 (4.00)
	Sigma-Ideal	253.53 (9.00)	213.50 (8.00)	243.00 (9.00)	179.57 (6.50)	163.07 (6.00)	175.73 (6.50)	75.07 (3.00)
	Sigma-MGA	245.73 (9.00)	239.77 (8.00)	253.60 (9.00)	173.67 (6.50)	110.50 (4.50)	226.83 (7.50)	73.37 (3.00)

Table 4.3: Kruskal-Wallis ranks for the $R2$ indicator

Obj.	Algorithms	DTLZ1	DTLZ2	DTLZ3	DTLZ4	DTLZ5	DTLZ6	DTLZ7
10	H-MOPSO	**19.67 (1.00)**	79.57 (3.00)	**21.07 (1.00)**	155.53 (5.50)	**26.53 (2.00)**	**45.40 (1.50)**	**19.97 (2.00)**
	SMPSO	258.50 (8.50)	**21.30 (2.50)**	263.17 (8.50)	**15.53 (1.50)**	96.80 (4.00)	110.83 (4.50)	229.30 (8.00)
	CD-Ideal	119.60 (4.00)	246.93 (8.50)	97.80 (4.00)	227.50 (8.00)	273.63 (9.00)	177.93 (6.00)	117.67 (4.00)
	CD-MGA	116.80 (4.00)	238.33 (8.50)	109.47 (4.00)	165.57 (6.00)	150.37 (5.50)	119.43 (5.00)	153.60 (5.50)
	NWSum-CD	96.73 (4.00)	93.27 (3.50)	136.77 (4.00)	77.50 (3.00)	164.37 (5.50)	172.97 (6.00)	201.87 (7.50)
	NWSum-Ideal	106.03 (4.00)	165.00 (5.50)	111.10 (4.00)	280.47 (9.00)	216.00 (7.50)	215.13 (7.00)	173.63 (7.00)
	NWSum-MGA	107.70 (4.00)	145.70 (5.00)	105.63 (4.00)	238.13 (8.50)	92.40 (4.00)	155.40 (6.00)	236.03 (8.00)
	Sigma-CD	208.47 (8.50)	**54.97 (2.50)**	226.60 (8.50)	110.73 (4.50)	132.60 (5.00)	204.40 (7.00)	210.10 (7.50)
	Sigma-Ideal	243.20 (8.50)	215.63 (7.50)	219.00 (8.50)	114.03 (4.50)	151.27 (5.50)	147.83 (6.00)	77.80 (2.50)
	Sigma-MGA	228.30 (8.50)	244.30 (8.50)	214.40 (8.50)	120.00 (4.50)	201.03 (7.00)	155.67 (6.00)	85.03 (3.00)
15	H-MOPSO	**22.60 (2.00)**	**79.43 (2.50)**	**19.07 (1.00)**	166.33 (5.50)	**29.83 (1.50)**	**25.00 (1.00)**	**18.73 (1.50)**
	SMPSO	259.80 (8.50)	**24.23 (2.50)**	281.30 (9.00)	**15.50 (1.50)**	109.77 (4.50)	116.73 (5.00)	220.77 (7.00)
	CD-Ideal	128.13 (4.00)	260.90 (8.50)	103.50 (4.00)	207.43 (7.50)	266.53 (9.00)	159.40 (5.50)	62.63 (2.00)
	CD-MGA	114.33 (4.00)	243.83 (8.50)	97.40 (4.00)	182.43 (7.00)	178.90 (6.50)	103.43 (5.00)	151.60 (6.50)
	NWSum-CD	112.10 (4.00)	95.40 (3.50)	134.70 (4.50)	97.27 (4.00)	164.40 (5.50)	206.60 (7.00)	172.47 (7.00)
	NWSum-Ideal	92.93 (3.50)	160.13 (5.50)	101.80 (4.00)	280.97 (9.50)	225.57 (8.00)	263.40 (9.50)	96.90 (3.00)
	NWSum-MGA	93.77 (3.50)	161.33 (6.00)	111.33 (4.00)	248.73 (8.50)	97.00 (3.50)	151.63 (5.50)	185.27 (7.00)
	Sigma-CD	208.27 (8.50)	**48.73 (2.50)**	207.33 (7.50)	76.30 (3.00)	104.90 (4.00)	160.50 (5.50)	204.27 (7.00)
	Sigma-Ideal	235.87 (8.50)	196.93 (7.50)	223.00 (8.50)	121.23 (4.50)	196.23 (7.50)	171.37 (5.50)	176.13 (7.00)
	Sigma-MGA	237.20 (8.50)	234.07 (8.00)	225.57 (8.50)	108.80 (4.00)	131.87 (5.00)	146.93 (5.50)	216.23 (7.00)
20	H-MOPSO	**16.53 (1.00)**	81.37 (2.50)	**21.30 (1.00)**	163.30 (5.50)	**20.70 (1.00)**	**22.27 (1.50)**	**15.53 (2.00)**
	SMPSO	272.03 (8.50)	**18.87 (2.00)**	268.13 (8.50)	**15.50 (1.50)**	94.90 (4.00)	123.93 (5.50)	194.27 (7.00)
	CD-Ideal	107.40 (4.00)	256.20 (8.50)	113.70 (4.00)	196.43 (7.50)	242.50 (8.50)	184.40 (6.00)	**46.23 (2.00)**
	CD-MGA	106.33 (4.00)	238.07 (8.50)	94.40 (4.00)	189.03 (7.00)	183.13 (7.00)	91.23 (3.50)	175.67 (6.50)
	NWSum-CD	123.33 (4.00)	97.03 (4.00)	121.77 (4.00)	118.73 (4.50)	141.83 (5.00)	183.30 (6.00)	153.20 (6.00)
	NWSum-Ideal	91.90 (4.00)	160.13 (5.50)	101.03 (4.00)	278.87 (9.50)	231.60 (8.50)	261.70 (10.00)	**75.27 (2.00)**
	NWSum-MGA	112.97 (4.00)	161.13 (5.50)	121.00 (4.00)	249.33 (8.50)	146.77 (5.00)	146.07 (5.50)	173.17 (6.50)
	Sigma-CD	217.73 (8.50)	50.20 (2.50)	220.03 (8.50)	83.83 (3.00)	107.90 (4.00)	163.60 (5.50)	161.00 (6.00)
	Sigma-Ideal	231.50 (8.50)	201.80 (7.50)	218.20 (8.50)	99.50 (4.00)	148.40 (5.00)	178.33 (6.00)	245.30 (8.00)
	Sigma-MGA	225.27 (8.50)	240.20 (8.50)	225.43 (8.50)	110.47 (4.00)	187.27 (7.00)	150.17 (5.50)	265.37 (9.00)

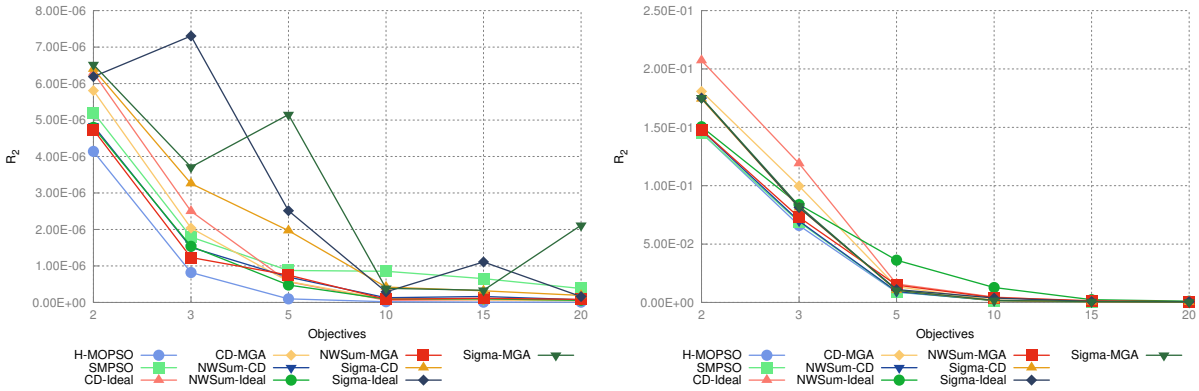
Table 4.4: Friedman overall ranks for the $R2$ indicator

H-MOPSO	SMPSO	CD-Ideal	CD-MGA	NWSum-CD	NWSum-Ideal	NWSum-MGA	Sigma-CD	Sigma-Ideal	Sigma-MGA
61.0 (1.0)	171.0 (4.5)	302.0 (7.5)	226.0 (5.0)	198.0 (4.5)	255.0 (6.0)	227.0 (5.0)	211.0 (4.5)	324.0 (8.5)	335.0 (8.5)



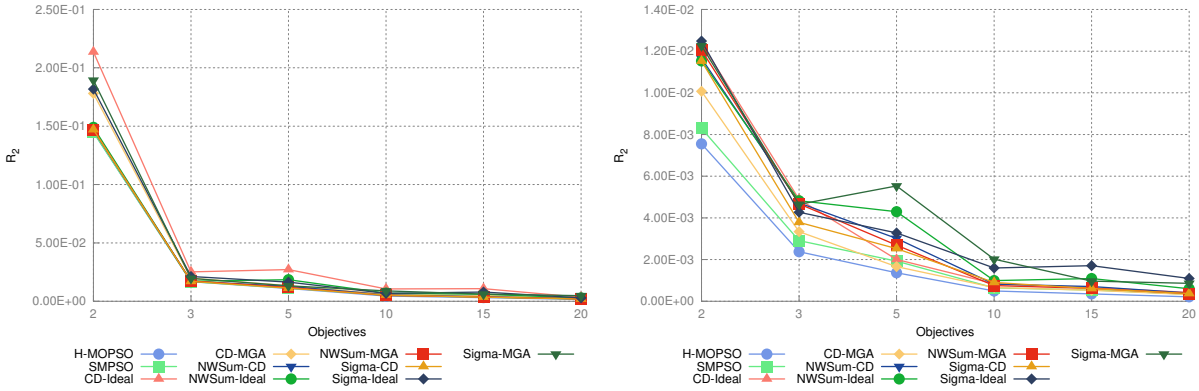
(a) DTLZ1 (b) DTLZ2

Figure 4.1: *R*2 results for problems DTLZ1 and DTLZ2.



(a) DTLZ3 (b) DTLZ4

Figure 4.2: *R*2 results for problems DTLZ3 and DTLZ4.



(a) DTLZ5 (b) DTLZ6

Figure 4.3: *R*2 results for problems DTLZ5 and DTLZ6.

From the presented results, for few objectives (two and three), H-MOPSO presents the general best performance, being among the best ranks in all the instances, except in DTLZ4 for two objectives. SMPSO (CD-CD) presented good results as well, outperforming H-MOPSO in DTLZ4 for two objectives, and being tied with it in six instances.

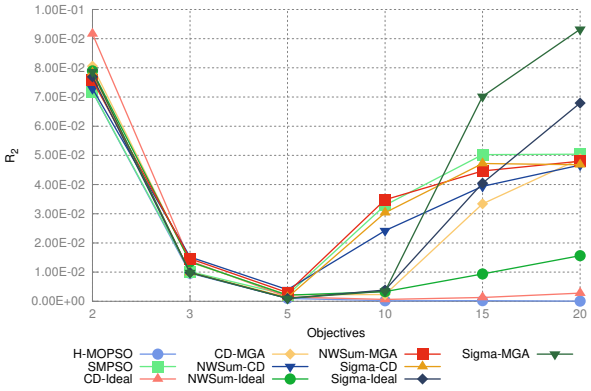


Figure 4.4: $R2$ results for problem DTLZ7.

For five objectives, H-MOPSO also presented the best general results, outperforming all the low-level heuristics in all problems, except for DTLZ4, where it statistically tied with SMPSO and NWSum-CD.

For ten objectives, again H-MOPSO had overall good results, losing only in DTLZ2 and DTLZ4. In DTLZ2, SMPSO and Sigma-CD had the best results, however in DTLZ4 only SMPSO achieved the smaller ranking.

For fifteen objectives, H-MOPSO had the best rankings in all problems, except for DTLZ4 where it lost, and in DTLZ2, where it tied with SMPSO and Sigma-CD. The best performance in DTLZ4 was achieved by SMPSO.

For twenty objectives, H-MOPSO also had the best performance in most problems, losing in DTLZ2 and DTLZ4, and being tied with CD-Ideal and NWSum-Ideal in DTLZ7. SMPSO had the best results in DTLZ2 and DTLZ4.

Regarding the performance of H-MOPSO per problem, in DTLZ1 it had alone the best rankings in all numbers of objectives, except for two, where it statistically tied with SMPSO. In DTLZ2, it was among the best performances in all cases, except for ten and twenty objectives. In DTLZ3, H-MOPSO had the best rankings alone in all cases.

DTLZ4 was the problem where H-MOPSO had the worst performance, being among the best algorithms only for three and five objectives. In DTLZ5, it shared the best rank with SMPSO for two and three objectives, and outperformed all the other algorithms for five objectives onwards.

In DTLZ6, H-MOPSO also shared the best ranking with SMPSO for two objectives, but outperformed all the other algorithms in the remaining cases. In DTLZ7, H-MOPSO shared the best results with SMPSO for two objectives, for twenty objectives, it statistically tied with CD-Ideal and NWSum-Ideal, outperforming the other algorithms in the remaining instances.

In general, H-MOPSO stood among the best algorithms in all the problems, except for some instances of DTLZ2 and DTLZ4. DTLZ2 is an easy problem that does not impose challenges to convergence or diversity, in this case the algorithms easily converge to the front, and increase the $R2$ value by improving the diversity. In this problem H-MOPSO was outperformed only by SMPSO and Sigma-CD that are two algorithms characterized by generating fronts with high diversity.

DTLZ4 is a hard problem by presenting diversity challenge. In this problem H-MOPSO had its worst performance, while SMPSO performed very well due to its high diversity characteristic.

Despite of the good results of SMPSO in few objectives and problems that demand higher diversity, it performs very badly in many-objective problems that present convergence challenge

like DTLZ1 and DTLZ3. The generality obtained by H-MOPSO due to its hyper-heuristic is an advantage in such cases.

Due to the high amount of data presented in Tables 4.2 and 4.3, it is hard to take overall conclusions from the performance of the algorithms over all the instances, hence we present Table 4.4 where the Friedman ranks obtained for the overall analysis of the algorithms are shown. In this test, the average of the 30 independent runs of each subproblem (problem/objective number) is considered. The test is performed with the 42 subproblems for each algorithm.

In the results summarized in this table, H-MOPSO achieves the lowest Friedman rank and the lowest final rank, which indicates that even without presenting the best individual result in all cases, it is a robust algorithm capable of obtaining good results in most of the cases.

Those results indicate that, in general, the proposed hyper-heuristic is able to properly select low-level heuristics to lead the search to regions that enhance its indicator values. The problems where H-MOPSO did not achieved the best performances in all cases (DTLZ2 and DTLZ4), are problems where the diversity characteristic is preferred over a balance of convergence and diversity, hence SMPSO, in general, obtained good results.

The next step is to analyze the behavior of the proposed algorithm through the probabilities of choosing the low-level heuristic along the search. Here we chose to show only the instances of two and twenty objectives, because they are representative of the others. The two objective instances were chosen because they have the smaller number of objectives, and the twenty objective instances were chosen because they have the largest number of objectives, hence represents the many-objective scenario.

Figures 4.5, 4.6, 4.7, 4.8, 4.9, 4.10 and 4.11 present the probabilities for the problems DTLZ1, DTLZ2, DTLZ3, DTLZ4, DTLZ5, DTLZ6 and DTLZ7 respectively. In these figures each line represents the average probability of choosing a low-level heuristic in a given iteration. The probabilities in each iteration are averaged over the 30 runs, to show a pattern.

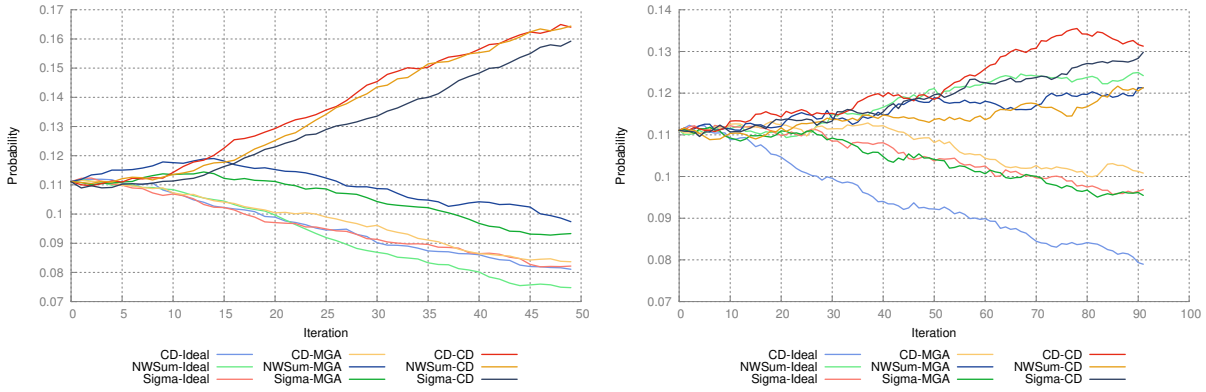
In these figures the number of iterations is different, because it is only shown the iterations in which the probabilities were updated, being ignored the first iterations where the SMPSO (CD-CD) algorithm was used until the repository is full.

For two objectives, in all cases, except for DTLZ2, the probability of selecting a low-level heuristic increases fast. In most cases (DTLZ1, DTLZ5, DTLZ6 and DTLZ7), the archiver Crowding Distance (CD) presented the better performance since its probabilities with any leader selection methods were higher than the others. In problem DTLZ4, the leader selection method seems to impact more, since the probabilities of the Sigma method with all the archivers had higher values. In DTLZ2 and DTLZ3 the probabilities of Sigma-CD, NWSum-CD and Sigma MGA were higher.

Considering the scenario of many-objective through the probabilities of selecting a low-level heuristic for the twenty objective instances, the behavior of the low-level heuristics was not so similar. The differences in the probabilities of selecting a heuristic, in general, took longer to increase.

In problem DTLZ1, despite of the greater time to achieve a larger difference in probabilities of selecting a heuristic, the combinations CD-CD and Sigma-CD had good chance of being selected, just like in two objectives, however the combination NWSum-CD that had good chances for two objectives, had its probabilities decreased for twenty objectives

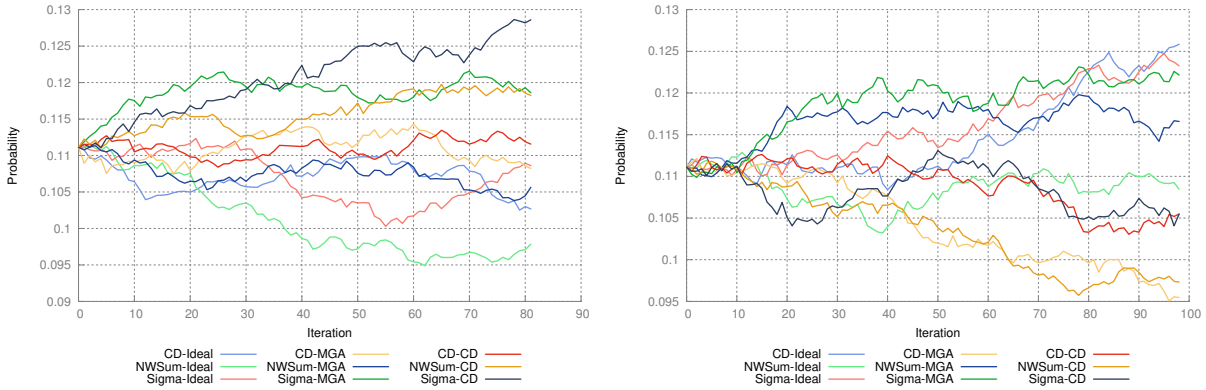
The behavior of the hyper-heuristic in problem DTLZ2 for twenty objectives was similar to its behavior for two objectives, with small differences in the probabilities during the entire search. The combination Sigma-MGA had good chances in both cases, but in two objectives the hyper-heuristic was seeking diversity, by improving the values of Sigma-CD and NWSum-CD,



(a) DTLZ1 - 2 Objectives

(b) DTLZ1 - 20 Objectives

Figure 4.5: Average probabilities over 30 runs for DTLZ1



(a) DTLZ2 - 2 Objectives

(b) DTLZ2 - 20 Objectives

Figure 4.6: Average probabilities over 30 runs for DTLZ2

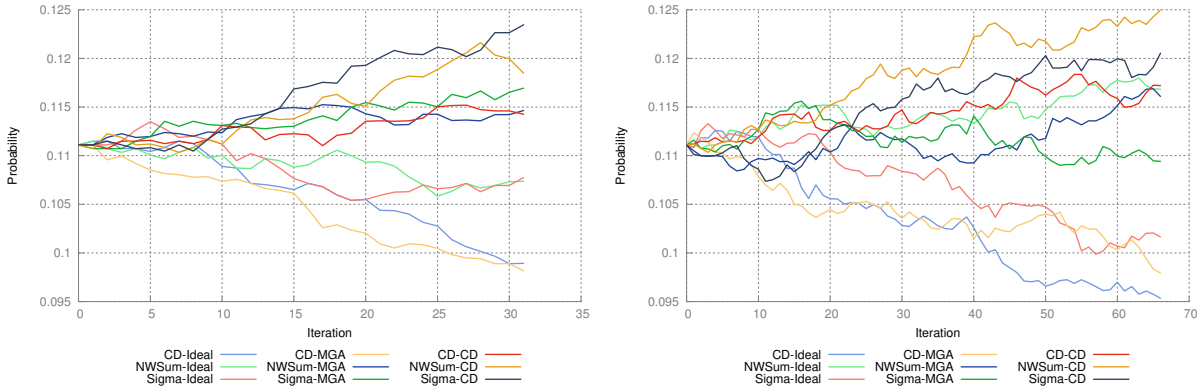
while in the twenty objective instance it was seeking convergence by using more often CD-Ideal and Sigma-Ideal.

In problem DTLZ3, the general behavior of the hyper-heuristic was similar to the two objective instance, however in this case the combination Sigma-MGA that had good performance for two objectives was replaced by NWSum-Ideal.

In general, the behavior of the hyper-heuristic in problem DTLZ4 for twenty objectives was similar to two. However, in two objectives the combinations Sigma-MGA and Sigma-Ideal had good performance while for twenty objectives they performed badly and were replaced by CD-CD and CD-MGA.

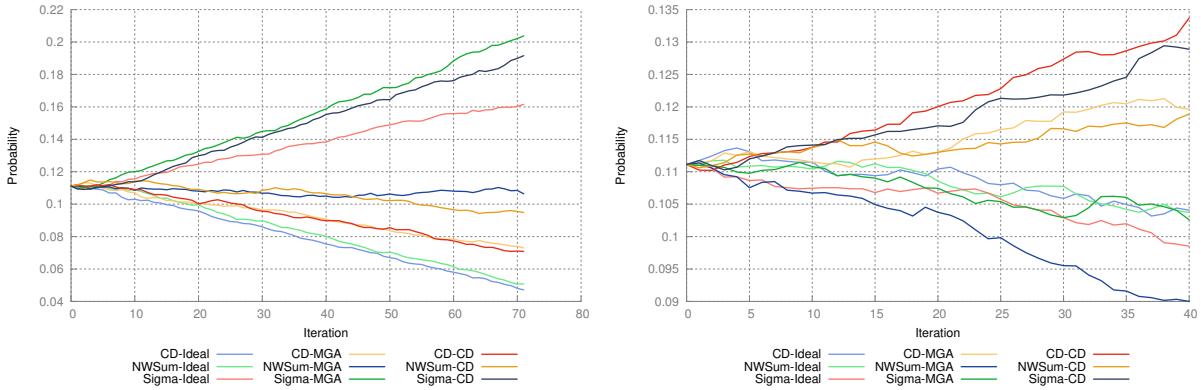
In problem DTLZ5, the hyper-heuristic had its probabilities very close, and the low-level heuristics that had the better performance in two objectives, performed worst in twenty. Most of the time the combinations presenting better performance were CD-Ideal, CD-MGA and NWSum-MGA .

Similarly to DTLZ5, in DTLZ6 the probabilities are also very close and present erratic behavior during the search and the combinations that perform well for two objectives had the worst results for twenty. In these problems the combinations CD-Ideal, CD-MGA and Sigma-MGA also had high probabilities.



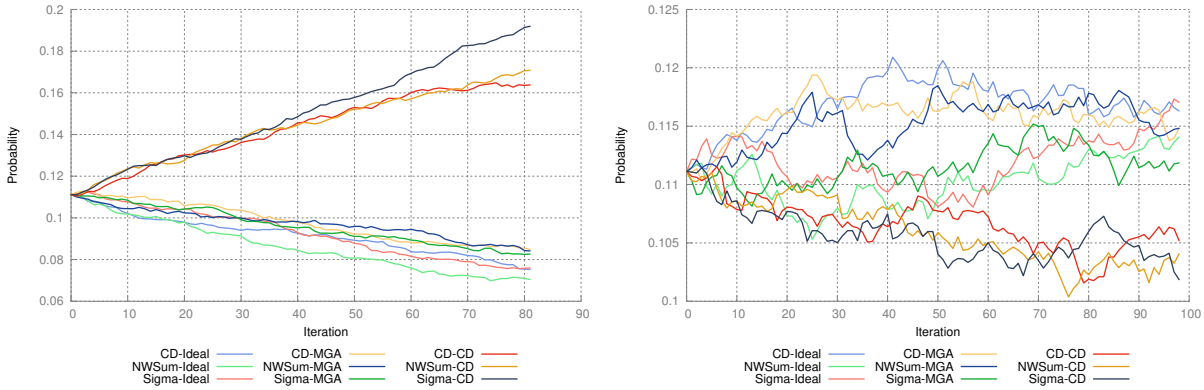
(a) DTLZ3 - 2 Objectives (b) DTLZ3 - 20 Objectives

Figure 4.7: Average probabilities over 30 runs for DTLZ3



(a) DTLZ4 - 2 Objectives (b) DTLZ4 - 20 Objectives

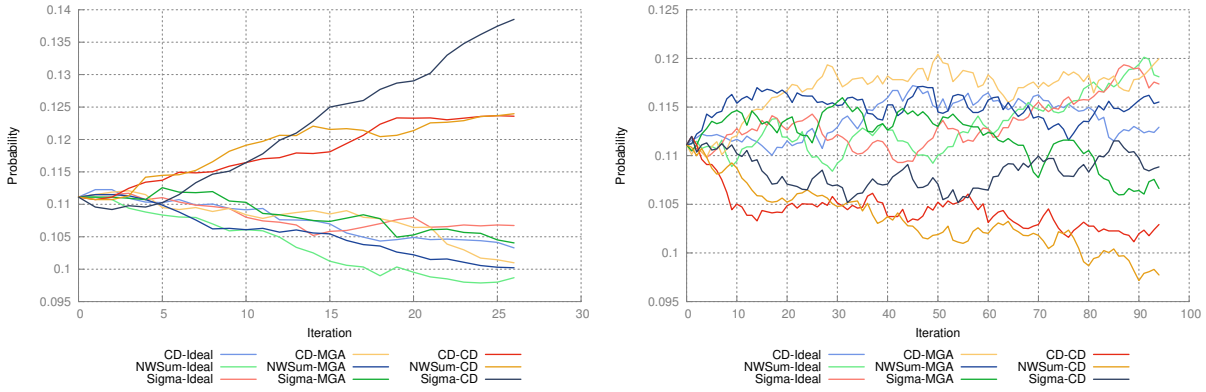
Figure 4.8: Average probabilities over 30 runs for DTLZ4



(a) DTLZ5 - 2 Objectives (b) DTLZ5 - 20 Objectives

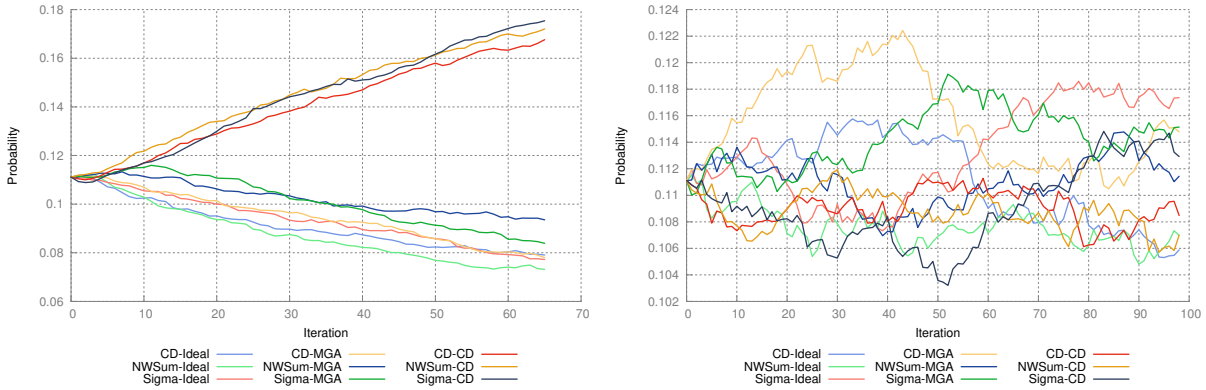
Figure 4.9: Average probabilities over 30 runs for DTLZ5

Problem DTLZ7 also had a behavior similar to DTLZ5 and DTLZ6, the combinations that performed well for two objectives had poor performance for twenty and the probabilities had erratic behavior. In the beginning of the search the combination CD-MGA had good performance,



(a) DTLZ6 - 2 Objectives (b) DTLZ6 - 20 Objectives

Figure 4.10: Average probabilities over 30 runs for DTLZ6



(a) DTLZ7 - 2 Objectives (b) DTLZ7 - 20 Objectives

Figure 4.11: Average probabilities over 30 runs for DTLZ7

but at the end, the hyper-heuristic increased the probabilities of the combination Sigma-Ideal to improve the convergence.

To indicate the compliance of the selection of the low-level heuristics with the probabilities, we observed the number of times (in the 30 executions) that each low-level heuristic was chosen. Those observations were done in three points, at approximately 1/3, 2/3 and at the end of the search for all the problems. These results are presented through Tables 4.5 and 4.6 for the instances of two and twenty objectives respectively.

Data in Tables 4.5 and 4.6 indicate that despite the selection mechanism does not always select the heuristics with the highest probability, those heuristics are usually among the most selected, especially closer to the end of the search, where the differences among the probabilities are higher.

When presenting the data from this section, we can comment about special features of our hyper-heuristic. One of them is the number of iterations needed by the roulette to present a great difference between the probabilities. In the beginning of the search, even a sub-optimal low-level heuristic can enhance the $R2$ results, because is natural for the search to find better solutions at first, however it can take several iterations to the roulette to increase the probabilities of the good heuristics.

Table 4.5: Number of times each heuristic was selected in the two objective instance of each problem

Iteration	DTLZ1								
	CD-Ideal	NWSum-Ideal	Sigma-Ideal	CD-MGA	NWSum-MGA	Sigma-MGA	CD-CD	NWSum-CD	Sigma-CD
It: (16)	1 (3.33%)	3 (10.00%)	4 (13.33%)	1 (3.33%)	5 (16.67%)	3 (10.00%)	8 (26.67%)	2 (6.67%)	3 (10.00%)
It: (33)	3 (10.00%)	3 (10.00%)	3 (10.00%)	4 (13.33%)	6 (20.00%)	3 (10.00%)	3 (10.00%)	1 (3.33%)	4 (13.33%)
It: (50)	1 (3.33%)	1 (3.33%)	2 (6.67%)	1 (3.33%)	0 (0.00%)	2 (6.67%)	7 (23.33%)	9 (30.00%)	7 (23.33%)
Iteration	DTLZ2								
It: (27)	2 (6.67%)	4 (13.33%)	2 (6.67%)	3 (10.00%)	2 (6.67%)	8 (26.67%)	3 (10.00%)	4 (13.33%)	2 (6.67%)
It: (54)	3 (10.00%)	1 (3.33%)	4 (13.33%)	6 (20.00%)	3 (10.00%)	0 (0.00%)	4 (13.33%)	5 (16.67%)	4 (13.33%)
It: (82)	2 (6.67%)	2 (6.67%)	3 (10.00%)	3 (10.00%)	3 (10.00%)	8 (26.67%)	3 (10.00%)	4 (13.33%)	2 (6.67%)
Iteration	DTLZ3								
It: (10)	2 (6.67%)	6 (20.00%)	0 (0.00%)	4 (13.33%)	1 (3.33%)	3 (10.00%)	5 (16.67%)	4 (13.33%)	4 (13.33%)
It: (21)	5 (16.67%)	2 (6.67%)	3 (10.00%)	5 (16.67%)	3 (10.00%)	3 (10.00%)	6 (20.00%)	2 (6.67%)	0 (0.00%)
It: (32)	4 (13.33%)	3 (10.00%)	1 (3.33%)	1 (3.33%)	3 (10.00%)	5 (16.67%)	5 (16.67%)	2 (6.67%)	5 (16.67%)
Iteration	DTLZ4								
It: (24)	2 (6.67%)	2 (6.67%)	4 (13.33%)	3 (10.00%)	5 (16.67%)	7 (23.33%)	2 (6.67%)	4 (13.33%)	1 (3.33%)
It: (48)	0 (0.00%)	3 (10.00%)	6 (20.00%)	3 (10.00%)	3 (10.00%)	6 (20.00%)	1 (3.33%)	2 (6.67%)	6 (20.00%)
It: (72)	0 (0.00%)	5 (16.67%)	6 (20.00%)	0 (0.00%)	1 (3.33%)	9 (30.00%)	3 (10.00%)	3 (10.00%)	3 (10.00%)
Iteration	DTLZ5								
It: (27)	2 (6.67%)	4 (13.33%)	1 (3.33%)	4 (13.33%)	2 (6.67%)	5 (16.67%)	6 (20.00%)	4 (13.33%)	2 (6.67%)
It: (54)	2 (6.67%)	2 (6.67%)	1 (3.33%)	4 (13.33%)	5 (16.67%)	1 (3.33%)	3 (10.00%)	7 (23.33%)	5 (16.67%)
It: (82)	2 (6.67%)	2 (6.67%)	2 (6.67%)	1 (3.33%)	2 (6.67%)	2 (6.67%)	9 (30.00%)	5 (16.67%)	5 (16.67%)
Iteration	DTLZ6								
It: (9)	3 (10.00%)	4 (13.33%)	8 (26.67%)	0 (0.00%)	1 (3.33%)	5 (16.67%)	3 (10.00%)	3 (10.00%)	2 (6.67%)
It: (18)	3 (10.00%)	5 (16.67%)	0 (0.00%)	2 (6.67%)	2 (6.67%)	3 (10.00%)	7 (23.33%)	4 (13.33%)	3 (10.00%)
It: (27)	2 (6.67%)	5 (16.67%)	2 (6.67%)	4 (13.33%)	5 (16.67%)	4 (13.33%)	4 (13.33%)	1 (3.33%)	2 (6.67%)
Iteration	DTLZ7								
It: (22)	3 (10.00%)	5 (16.67%)	5 (16.67%)	3 (10.00%)	2 (6.67%)	2 (6.67%)	3 (10.00%)	3 (10.00%)	4 (13.33%)
It: (44)	2 (6.67%)	3 (10.00%)	0 (0.00%)	3 (10.00%)	1 (3.33%)	6 (20.00%)	4 (13.33%)	5 (16.67%)	6 (20.00%)
It: (66)	2 (6.67%)	1 (3.33%)	3 (10.00%)	3 (10.00%)	1 (3.33%)	2 (6.67%)	10 (33.33%)	4 (13.33%)	4 (13.33%)

Table 4.6: Number of times each heuristic was selected in the twenty objective instance of each problem

Iteration	DTLZ1								
	CD-Ideal	NWSum-Ideal	Sigma-Ideal	CD-MGA	NWSum-MGA	Sigma-MGA	CD-CD	NWSum-CD	Sigma-CD
It: (30)	2 (6.67%)	7 (23.33%)	3 (10.00%)	3 (10.00%)	4 (13.33%)	2 (6.67%)	5 (16.67%)	2 (6.67%)	2 (6.67%)
It: (61)	1 (3.33%)	5 (16.67%)	4 (13.33%)	4 (13.33%)	4 (13.33%)	5 (16.67%)	3 (10.00%)	2 (6.67%)	2 (6.67%)
It: (92)	1 (3.33%)	6 (20.00%)	1 (3.33%)	5 (16.67%)	1 (3.33%)	2 (6.67%)	3 (10.00%)	4 (13.33%)	7 (23.33%)
Iteration	DTLZ2								
It: (33)	3 (10.00%)	2 (6.67%)	3 (10.00%)	2 (6.67%)	5 (16.67%)	6 (20.00%)	3 (10.00%)	4 (13.33%)	2 (6.67%)
It: (66)	2 (6.67%)	2 (6.67%)	4 (13.33%)	4 (13.33%)	3 (10.00%)	8 (26.67%)	3 (10.00%)	2 (6.67%)	2 (6.67%)
It: (99)	5 (16.67%)	2 (6.67%)	2 (6.67%)	5 (16.67%)	4 (13.33%)	3 (10.00%)	2 (6.67%)	4 (13.33%)	3 (10.00%)
Iteration	DTLZ3								
It: (22)	3 (10.00%)	4 (13.33%)	7 (23.33%)	3 (10.00%)	2 (6.67%)	4 (13.33%)	1 (3.33%)	3 (10.00%)	3 (10.00%)
It: (44)	3 (10.00%)	3 (10.00%)	6 (20.00%)	0 (0.00%)	3 (10.00%)	2 (6.67%)	4 (13.33%)	5 (16.67%)	4 (13.33%)
It: (67)	1 (3.33%)	3 (10.00%)	3 (10.00%)	5 (16.67%)	6 (20.00%)	0 (0.00%)	4 (13.33%)	6 (20.00%)	2 (6.67%)
Iteration	DTLZ4								
It: (13)	3 (10.00%)	2 (6.67%)	2 (6.67%)	7 (23.33%)	2 (6.67%)	3 (10.00%)	6 (20.00%)	2 (6.67%)	3 (10.00%)
It: (27)	4 (13.33%)	3 (10.00%)	6 (20.00%)	4 (13.33%)	4 (13.33%)	3 (10.00%)	4 (13.33%)	1 (3.33%)	1 (3.33%)
It: (41)	3 (10.00%)	1 (3.33%)	4 (13.33%)	7 (23.33%)	3 (10.00%)	3 (10.00%)	3 (10.00%)	2 (6.67%)	4 (13.33%)
Iteration	DTLZ5								
It: (33)	2 (6.67%)	3 (10.00%)	4 (13.33%)	1 (3.33%)	6 (20.00%)	5 (16.67%)	3 (10.00%)	4 (13.33%)	2 (6.67%)
It: (66)	2 (6.67%)	4 (13.33%)	2 (6.67%)	4 (13.33%)	3 (10.00%)	7 (23.33%)	4 (13.33%)	3 (10.00%)	1 (3.33%)
It: (99)	4 (13.33%)	2 (6.67%)	4 (13.33%)	4 (13.33%)	4 (13.33%)	2 (6.67%)	3 (10.00%)	4 (13.33%)	3 (10.00%)
Iteration	DTLZ6								
It: (31)	0 (0.00%)	1 (3.33%)	3 (10.00%)	7 (23.33%)	8 (26.67%)	3 (10.00%)	3 (10.00%)	2 (6.67%)	3 (10.00%)
It: (63)	4 (13.33%)	2 (6.67%)	0 (0.00%)	4 (13.33%)	6 (20.00%)	5 (16.67%)	2 (6.67%)	5 (16.67%)	2 (6.67%)
It: (95)	5 (16.67%)	3 (10.00%)	2 (6.67%)	2 (6.67%)	4 (13.33%)	2 (6.67%)	1 (3.33%)	2 (6.67%)	9 (30.00%)
Iteration	DTLZ7								
It: (33)	2 (6.67%)	2 (6.67%)	4 (13.33%)	2 (6.67%)	5 (16.67%)	6 (20.00%)	2 (6.67%)	5 (16.67%)	2 (6.67%)
It: (66)	1 (3.33%)	5 (16.67%)	2 (6.67%)	4 (13.33%)	3 (10.00%)	7 (23.33%)	4 (13.33%)	3 (10.00%)	1 (3.33%)
It: (99)	3 (10.00%)	3 (10.00%)	3 (10.00%)	5 (16.67%)	3 (10.00%)	3 (10.00%)	3 (10.00%)	3 (10.00%)	4 (13.33%)

Other problem noted is related to the roulette mechanism. When using a high number of low-level heuristics, if they present great differences in performance, as in Figure 4.9(a), better heuristics are selected more frequently and the chances of getting better results are higher.

However when the differences in the probabilities are smaller, as in Figure 4.11(b), sub-optimal heuristics are selected frequently, hence sub-optimal results are expected with the hyper-heuristic.

Another problem observed is regarding the $R2$ indicator that in some problems with special Pareto shapes, like discontinuous or degenerate, can lead to misleading results. The focus of future works will be addressing such problems.

4.2.3 Heuristic selection methods

In this section, we compare the four heuristic selection methods (FRRMAB, ACF, random and roulette) to identify which is the best when used within the H-MOPSO. The parameters were set following (Li et al., 2014), where the numbers of objectives used were: 2, 3, 5, 8 and 10. The population and repository maximum size were set according to the number of objectives as 91, 91, 210, 156 and 275, respectively. The maximum number of iterations was set according to the problem and the objective number, as presented in Table 4.7. A representative set of benchmark problems from the DTLZ (Deb et al., 2002) and WFG (Huband et al., 2006) benchmark families was used, this set is composed of problems DTLZ1 to DTLZ4, WFG6 and WFG7.

Table 4.7: Number of iterations per problem instance.

Problem	**Objective Number**				
	2	**3**	**5**	**8**	**10**
DTLZ1	400	400	600	750	1000
DTLZ2	250	250	350	500	750
DTLZ3	1000	1000	1000	1000	1500
DTLZ4	600	600	1000	1250	2000
WFG6	400	400	750	1500	2000
WFG7	400	400	750	1500	2000

Table 4.8: IGD: Mean and standard deviation for Random, ACF, FRRMAB, and Roulette

Obj.	Problem	FRRMAB	ACF	RANDOM	ROULETTE
2	DTLZ1	1.32E-3(1.65E-3)	1.52E-3(1.66E-3)	**1.12E-3(1.45E-3)**	2.54E-3(9.26E-3)
	DTLZ2	6.75E-4(1.35E-4)	7.37E-4(1.17E-4)	7.06E-4(1.2E-4)	**6.54E-4(6.24E-5)**
	DTLZ3	**6.43E-4(1.63E-4)**	5.97E-3(1.51E-2)	3.2E-3(1.1E-2)	3.21E-3(1.1E-2)
	DTLZ4	6.02E-4(6.88E-5)	6.96E-4(1.64E-4)	6.03E-4(1.15E-4)	**5.83E-4(4.4E-5)**
	WFG6	3.91E-3(5.27E-3)	3.66E-3(4.13E-3)	5.85E-3(6.28E-3)	**1.61E-3(5.93E-4)**
	WFG7	3.69E-3(2.39E-3)	2.61E-3(1.26E-3)	3.18E-3(1.78E-3)	**1.92E-3(3.6E-4)**
3	DTLZ1	8.58E-3(5.99E-3)	6.83E-3(3.75E-4)	7.45E-3(2.48E-3)	**6.69E-3(5.11E-4)**
	DTLZ2	**9.52E-3(6.4E-4)**	9.56E-3(5.87E-4)	9.57E-3(7.15E-4)	9.66E-3(1.01E-3)
	DTLZ3	1.82E-2(1.7E-2)	**9.83E-3(2.19E-3)**	1.41E-2(1.14E-2)	1.35E-2(1.1E-2)
	DTLZ4	**8.96E-3(7.3E-4)**	9.34E-3(1.2E-3)	9.4E-3(9.8E-4)	1.14E-2(3.23E-3)
	WFG6	1.12E-2(1.7E-3)	1.06E-2(1.41E-3)	1.14E-2(2.19E-3)	**1E-2(8.04E-4)**
	WFG7	1.49E-2(2.22E-3)	1.51E-2(2.06E-3)	1.55E-2(2.29E-3)	**1.3E-2(4.86E-4)**
5	DTLZ1	1.22E-2(1.02E-3)	1.19E-2(6.24E-4)	1.15E-2(5.01E-4)	**1.08E-2(5.93E-4)**
	DTLZ2	1.77E-2(6.58E-4)	1.77E-2(7.89E-4)	1.76E-2(6.85E-4)	**1.71E-2(9.74E-4)**
	DTLZ3	**1.77E-2(7.92E-4)**	1.85E-2(1.02E-3)	1.79E-2(8.37E-4)	1.98E-2(3.53E-3)
	DTLZ4	**1.54E-2(4.04E-4)**	1.55E-2(4.73E-4)	1.55E-2(5.61E-4)	1.68E-2(1.6E-3)
	WFG6	1.59E-2(7.91E-4)	1.57E-2(8.16E-4)	1.59E-2(5.25E-4)	**1.55E-2(8.82E-4)**
	WFG7	1.92E-2(1.12E-3)	1.86E-2(6.25E-4)	1.93E-2(1.14E-3)	**1.83E-2(8.68E-4)**
8	DTLZ1	2.55E-2(1.57E-3)	2.52E-2(1.7E-3)	2.61E-2(1.91E-3)	**2.47E-2(1.44E-3)**
	DTLZ2	4.79E-2(2.77E-3)	4.78E-2(2.44E-3)	**4.71E-2(2.69E-3)**	5.08E-2(3.72E-3)
	DTLZ3	5.37E-2(3.61E-3)	**5.14E-2(3.87E-3)**	5.41E-2(5.43E-3)	5.35E-2(5.13E-3)
	DTLZ4	**3.54E-2(2.23E-3)**	3.6E-2(2.86E-3)	3.57E-2(3.13E-3)	3.84E-2(2.53E-3)
	WFG6	4.37E-2(4E-3)	**4.14E-2(2.81E-3)**	4.16E-2(3.71E-3)	4.36E-2(3.67E-3)
	WFG7	4.57E-2(3.12E-3)	4.52E-2(2.95E-3)	**4.51E-2(3.15E-3)**	4.75E-2(3.98E-3)
10	DTLZ1	2.06E-2(9.98E-4)	2.03E-2(8.68E-4)	2.03E-2(1.03E-3)	**1.93E-2(9.84E-4)**
	DTLZ2	4.16E-2(2.27E-3)	4.07E-2(2.74E-3)	**4.07E-2(2.37E-3)**	4.46E-2(1.8E-3)
	DTLZ3	4.35E-2(2.73E-3)	**4.18E-2(2E-3)**	4.3E-2(2.54E-3)	4.45E-2(4.29E-3)
	DTLZ4	2.83E-2(1.31E-3)	**2.72E-2(1.22E-3)**	2.73E-2(1.34E-3)	2.97E-2(2.2E-3)
	WFG6	**3.63E-2(2.65E-3)**	3.68E-2(3.29E-3)	3.76E-2(3.46E-3)	3.82E-2(2.73E-3)
	WFG7	**5.78E-2(3.46E-3)**	5.93E-2(4.77E-3)	5.98E-2(3.07E-3)	6.46E-2(4.88E-3)

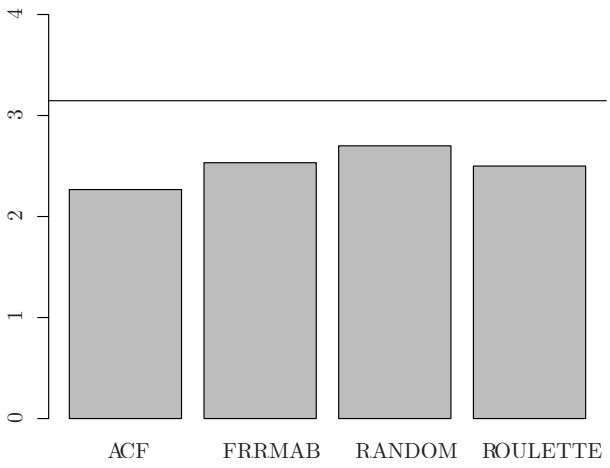


Figure 4.12: Average ranking for FRRMAB, ACF, Roulette, and Random for the IGD indicator

Table 4.9: HV: Mean and standard deviation for Random, ACF, FRRMAB, and Roulette

Obj.	Problem	FRRMAB	ACF	RANDOM	ROULETTE
2	DTLZ1	9.97E-1(3.21E-3)	9.97E-1(3.4E-3)	**9.98E-1(2.95E-3)**	9.97E-1(8.34E-3)
	DTLZ2	2.09E-1(7.5E-4)	2.09E-1(9.41E-4)	2.09E-1(6.97E-4)	**2.1E-1(2.23E-4)**
	DTLZ3	**1E0(1.1E-7)**	9.99E-1(3.06E-3)	9.99E-1(4.11E-3)	9.99E-1(1.43E-3)
	DTLZ4	2.1E-1(5.05E-4)	2.09E-1(1.3E-3)	2.1E-1(9.42E-4)	**2.1E-1(1.79E-4)**
	WFG6	2.46E-1(9.53E-3)	2.45E-1(8.79E-3)	2.43E-1(1.16E-2)	**2.49E-1(5.65E-3)**
	WFG7	2.2E-1(4.97E-3)	2.22E-1(4.8E-3)	2.17E-1(6.33E-3)	**2.25E-1(5.16E-3)**
3	DTLZ1	9.97E-1(4.14E-3)	9.99E-1(9.19E-4)	9.99E-1(1.08E-3)	**1E0(1.06E-5)**
	DTLZ2	7.73E-1(2.59E-3)	7.69E-1(8.95E-3)	7.72E-1(2.59E-3)	**7.75E-1(2.75E-3)**
	DTLZ3	1E0(1.18E-4)	**1E0(3.21E-6)**	1E0(6.74E-6)	1E0(1.23E-5)
	DTLZ4	**4.83E-1(4.3E-3)**	4.83E-1(3.89E-3)	4.82E-1(5.57E-3)	4.82E-1(6.63E-3)
	WFG6	3.78E-1(8.25E-3)	3.8E-1(9.46E-3)	3.78E-1(1.37E-2)	**3.93E-1(7.87E-3)**
	WFG7	3.87E-1(1.89E-2)	3.82E-1(1.51E-2)	3.85E-1(1.62E-2)	**4.13E-1(1.05E-2)**
5	DTLZ1	9.99E-1(2.1E-3)	9.99E-1(1.3E-3)	9.99E-1(1.48E-3)	**1E0(5.92E-5)**
	DTLZ2	9.89E-1(2.54E-3)	9.9E-1(5.7E-4)	9.9E-1(7.14E-4)	**9.9E-1(3.02E-3)**
	DTLZ3	1E0(0E0)	**1E0(0E0)**	1E0(0E0)	1E0(1.12E-10)
	DTLZ4	9.9E-1(7.3E-4)	9.9E-1(3.31E-4)	9.9E-1(7.72E-4)	**9.9E-1(7.43E-4)**
	WFG6	5.4E-1(2.25E-2)	5.44E-1(3.27E-2)	5.32E-1(2.24E-2)	**5.77E-1(1.43E-2)**
	WFG7	5E-1(1.72E-2)	5.09E-1(1.56E-2)	4.97E-1(1.6E-2)	**5.39E-1(1.74E-2)**
8	DTLZ1	1E0(3.01E-7)	1E0(1.15E-6)	1E0(1.17E-7)	**1E0(3.62E-8)**
	DTLZ2	9.97E-1(9.16E-4)	9.97E-1(1.66E-3)	9.97E-1(2.04E-3)	**9.97E-1(1.1E-3)**
	DTLZ3	1E0(0E0)	**1E0(0E0)**	1E0(2.24E-11)	1E0(0E0)
	DTLZ4	1E0(7.05E-5)	1E0(3.41E-5)	1E0(5.86E-5)	**1E0(3.25E-5)**
	WFG6	5.43E-1(4.75E-2)	5.64E-1(3.61E-2)	5.69E-1(4.47E-2)	**5.73E-1(5.25E-2)**
	WFG7	4.3E-1(3.23E-2)	4.44E-1(3.35E-2)	**4.55E-1(4.18E-2)**	4.42E-1(2.91E-2)
10	DTLZ1	**1E0(3.42E-10)**	1E0(1.14E-9)	1E0(1.25E-9)	1E0(3.32E-10)
	DTLZ2	9.98E-1(1.3E-3)	**9.99E-1(7.56E-4)**	9.98E-1(9.83E-4)	9.98E-1(1.22E-3)
	DTLZ3	1E0(0E0)	**1E0(0E0)**	1E0(0E0)	1E0(3.08E-11)
	DTLZ4	1E0(5.05E-6)	1E0(4.15E-6)	1E0(6.59E-6)	**1E0(2.77E-6)**
	WFG6	6.48E-1(4.95E-2)	6.43E-1(4.7E-2)	6.46E-1(4.5E-2)	**6.79E-1(5.68E-2)**
	WFG7	4.62E-1(1.94E-2)	4.68E-1(3.32E-2)	**4.72E-1(3.61E-2)**	4.71E-1(4.11E-2)

The average IGD value of each algorithm for each problem and number of objectives is detailed in Table 4.8. The best results are highlighted in bold font, and if there are statistically significant differences according to the Kruskal-Wallis test, the best algorithm is highlighted with the gray background. Since non-parametric statistical tests, like the Kruskal-Wallis test, are conservative, we draw conclusions based on the average indicator values when significant differences are not found.

According to the IGD indicator, the best average values are distributed among the algorithms. However if we consider the number of instances (problem/objective number) where each algorithm achieved the best average results, it can be seen that Roulette performed better

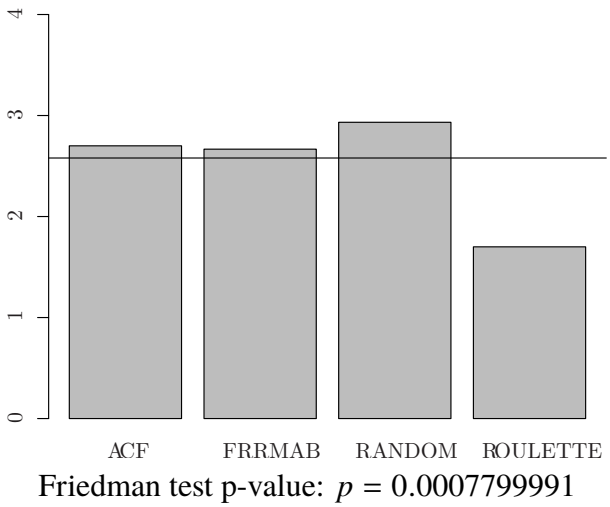


Figure 4.13: Average ranking for FRRMAB, ACF, Roulette, and Random for the HV indicator

than the others, followed by FRRMAB, then ACF and lastly RANDOM. If we consider only the statistical differences found, then Roulette was the only one that significantly outperformed the others, but only on WFG7 for three objectives and DTLZ1 for ten objectives.

To facilitate the visualization of the data presented in Table 4.8 and to allow drawing general conclusions, we considered the performance per algorithm, ignoring particular problems and objective numbers. To do so, we calculated the average of the 20 independent runs of each algorithm for all problems and objective numbers and used these averages as samples in the Friedman test; the results are presented at Figure 4.12. In this figure, each bar represents the average ranking of an algorithm (the smaller, the better) and the horizontal line represents the threshold to achieve statistically significant difference to the best performing algorithm, i.e., the ranking of the best plus the critical difference. This line means that all the algorithms whose mean rankings are below it are considered statistically equivalent to the best.

This test was not able to identify significant differences between any algorithms. However, the best average result was obtained by ACF. Despite being the best algorithm with a statistical difference for some problems, in this general comparison Roulette had the second best average performance.

The average results considering the Hypervolume of each algorithm for each problem and objective number are presented in Table 4.9. Regarding these results, Roulette had the best average performance with a statistical difference in ten of thirty problem instances. Moreover, it presented the best average results without a statistical difference in other nine instances, totaling best results in nineteen instances. The second best algorithm is ACF, with best average results in five instances, followed by FRRMAB and Random respectively.

As done for the IGD, we also summarized the results per algorithm for the hypervolume in Figure 4.13. In this case, the best mean hypervolume was achieved by Roulette, where it obtained significant statistical difference when compared to all other heuristic selection methods evaluated (Random, ACF, and FRRMAB) according to the Friedman statistical test.

The results presented indicate that the Roulette selection method is the best for the H-MOPSO algorithm. The methods FRRMAB and ACF achieved similar results in general, and presented superior results than the Random, although no statistically significant differences were found.

This result confirms our hypothesis that the heuristic selection method has an impact on the performance of the H-MOPSO algorithm; however, this impact was weaker than anticipated. Moreover, we expected that the state-of-the-art heuristic selection methods would outperform the simple roulette wheel. However, this was not the case.

4.2.4 H-MOPSO-Roulette parameter configuration

In the previous comparative study, we identified that the Roulette was the best performing heuristic selection method. Hence in this section, we conduct an additional study on it to further improve its performance. To do so, a new parameter is introduced and calibrated. This new parameter (K) is used to set how many times a single low-level heuristic will be employed before another one is selected. In the previous versions of the H-MOPSO, the value of K was one, which means that every iteration a new low-level heuristic was selected, with the possibility of selecting the same.

For the sensitivity analysis of the parameter K, we selected the problem DTLZ3 for three objectives because of the poor results found by H-MOPSO-Roulette on this problem. The results obtained with different K values were evaluated using the IGD indicator.

The current study is divided in three. First, we calibrate the parameter K only for selecting the archiving method, with the leader selection fixed. Then we conduct a similar study only for the leader selection method, with the archiving fixed. Finally, we conduct another study calibrating K for both approaches at the same time.

At the end of the section, we compare the best results obtained in each of the three sensitivity studies to identify the best setting.

4.2.4.1 Calibrating the archiving selection interval

In this study, we configure K for selecting only the archiving method. Hence we keep the leader selection method fixed, to always use the SMPSO default (Crowding Distance). The values used in this study were K={1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200, 300, 400, 500}. For example, if $K = 70$ the selected low-level heuristic (archive method) will be executed 70 times before being replaced.

The Figure 4.14 presents the IGD results of this comparison, through boxplots for each K value. Among the compared values, the best results were obtained with $K = 30$, while $K = 300$ achieved the worst results in general.

A Kruskal-Wallis statistical test was performed between the results found with each K value, but significant statistical differences were only found between $K = \{30, 40, 50\}$, which statistically outperformed $K = 300$, as can be seen in Table 4.10. These results indicate that values near the optimum ($K = 30$) perform well.

Table 4.10: Multiple comparison test after Kruskal-Wallis (When difference is TRUE)

Comparison		Obs. diff.	Critical diff.
K=30	K=300	133.066667	117.3159
K=300	**K=40**	124.066667	117.3159
K=300	**K=50**	122.266667	117.3159

4.2.4.2 Calibrating the leader selection group size

A similar study was performed, this time for the leader selection method. In this study, the archiving method was fixed to only use the SMPSO default: Crowding Distance.

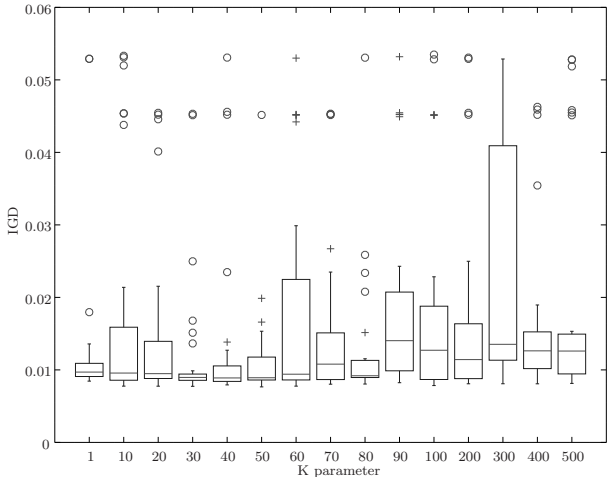


Figure 4.14: Configuration of the parameter K for Archiving

Unlike the archiving method, a different leader selection method can be attributed to each particle, which means that we can have groups of different sizes using the same leader selection method, from 1 to the whole population. Hence, the K parameter here means the number of particles (group size) in which the same leader selection method was used. In this study the values investigated are K={1, 11, 21, 31, 41, 51, 61, 71, 81, 91}.

The IGD results of this study are presented on Figure 4.15 for each K value. In general, the best performing K value was 91, which is equal to the population size. This means that using the same leader selection method for all the population is the best, however, no statistically significant differences were found in this study according to the Kruskal-Wallis test.

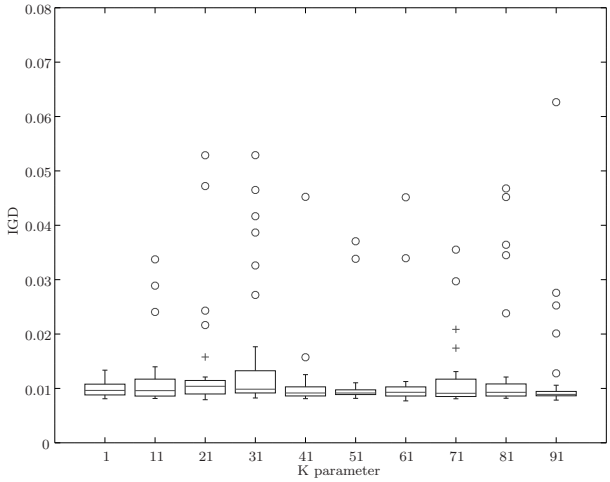


Figure 4.15: Configuration of the parameter K for Leader Selection

4.2.4.3 Calibrating leader and archiving selection interval

A third experimental study was conducted, where both methods (Archive and Leader Selection) were simultaneously replaced after K iterations. Our goal here is to investigate which is the best combination of leader and archiving methods since the optimal value of the method can weight more to the final optimal configuration of the framework. Hence it is very important

to evaluate them together. Moreover, by selecting both methods together, we expect to have a stronger contribution of the low-level heuristics in the final results.

The values evaluated were K={1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200, 300, 400, 500}, as in the first study.

The IGD results of this combined study can be appreciated on Figure 4.16. Here we can see that, in general, smaller values are better than the larger, and the best value is $K = 1$ as in the original H-MOPSO. Statistical differences were found between $K = 1$ and K=90, 100, 300, 400, 500 according to the Kruskal-Wallis test. The results of this test are presented in Table 4.11.

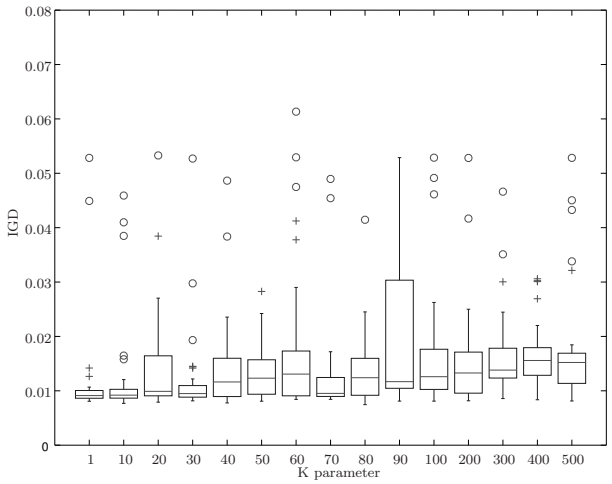


Figure 4.16: Configuration of the parameter K for Archive and Leader Selection

Table 4.11: Multiple comparison test after Kruskal-Wallis (When difference is TRUE)

Comparison		**Obs. diff.**	**Critical diff.**
K=1	K=100	117.93	117.32
K=1	K=300	144.93	117.32
K=1	K=400	157.37	117.32
K=1	K=500	148.77	117.32
K=1	K=90	133.80	117.32
K=10	K=300	127.30	117.32
K=10	K=400	139.73	117.32
K=10	K=500	131.13	117.32
K=30	K=300	119.57	117.32
K=30	K=400	132.00	117.32
K=30	K=500	123.40	117.32
K=400	**K=70**	117.63	117.32

4.2.4.4 Comparison between selecting leader, archiving or both simultaneously

In previous sections, three different experimental studies were conducted. First, we investigated the best value for K to select the archiving method and found $K = 30$ as the best value. Next, we investigated how many particles in the population should use the same leader selection method; the best value obtained suggest that updating the entire population with the same method is best. Finally, we combined both methods to investigate which is the best interval for the pair leader selection / archiving; this study indicated that changing each $K = 1$ iterations is the best, as was originally done in the H-MOPSO algorithm.

In this section, we compare the best configurations obtained in each of the three sections, to determine the best configuration for H-MOPSO-Roulette. The results of this comparison are presented in Figure 4.17 and Table 9.

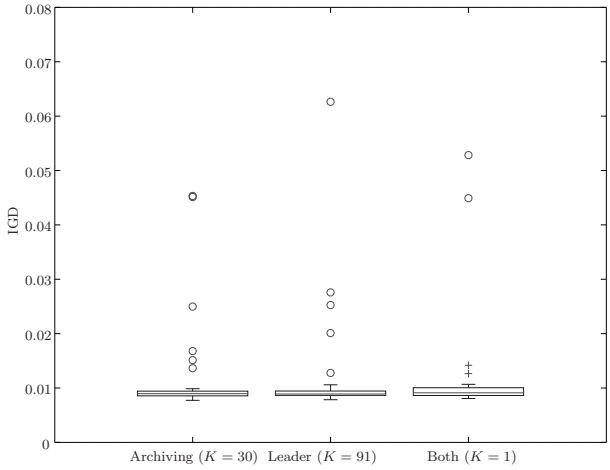


Figure 4.17: IGD comparison between selecting leader, archiving or both simultaneously

Table 4.12: IGD comparison between selecting leader, archiving or both simultaneously

Obj.	Problem	Archiving ($K = 30$)	Leader ($K = 91$)	Both ($K = 1$)
3	DTLZ3	1,24E-2(9,59E-3)	1,23E-2(1,07E-2)	**1,21E-2(1,01E-2)**

Based on the presented results, we can conclude that simultaneously selecting both methods seems slightly better than selecting only one. However, no statistically significant difference was found.

4.2.5 H-MOPSO-Roulette vs. MOEA/D-FRRMAB

In the previous sections, we presented a comparison between heuristic selection methods. Next, we selected the Roulette, since it was the best performing heuristic selection method and conducted a sensitivity analysis of an important parameter: the interval for changing the low-level heuristic.

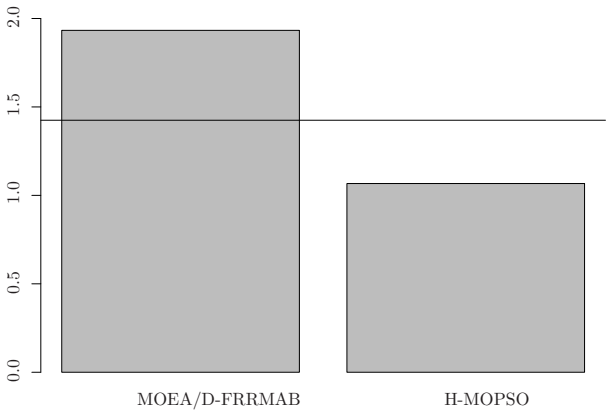
In this section, our goal is to validate the performance of our framework against a state-of-the-art hyper-heuristic algorithm. To do so, we selected the MOEA/D-FRRMAB (Li et al., 2014), since as H-MOPSO, it employs a hyper-heuristic as a component to improve its performance.

The parameters used here for H-MOPSO are the same used in Section 4.2.3. For MOEA/D-FRRMAB we followed the same parameters used by Li et al. (2014), where the neighborhood size T was set to $0.1 \times |\boldsymbol{W}|$ and the maximum number of solutions replaced by each child solution n_r was set to $0.01 \times |\boldsymbol{W}|$, where $|\boldsymbol{W}|$ is the number of subproblems. The probability of selecting a parent from the neighborhood δ is set to 0.9 and the control parameters for the differential evolution (DE) are $CR = 1$ and $F = 0.5$. The parameters for the polynomial mutation are: distribution index $\eta = 20$ and mutation rate $p_m = 1/n$ where n is the number of decision variables.

When evaluating the results obtained by the IGD indicator presented in Table 4.13, H-MOPSO-ROULETTE had the best value with significant statistical difference in most of the instances. H-MOPSO-ROULETTE was outperformed with a statistical difference only on DTLZ2 and WFG7 for three objectives.

Table 4.13: IGD: Mean (and standard deviation) for H-MOPSO-ROULETTE and MOEA/D-FRRMAB

Obj.	Problem	H-MOPSO-ROULETTE	MOEA/D-FRRMAB
2	DTLZ1	**2.31E-3(7.28E-3)**	4.08E-2(2.26E-3)
	DTLZ2	**6.65E-4(6.82E-5)**	4.98E-2(5.48E-8)
	DTLZ3	**3.05E-3(1.1E-2)**	4.98E-2(6.94E-6)
	DTLZ4	**5.92E-4(3.59E-5)**	4.98E-2(6.04E-8)
	WFG6	**1.53E-3(6.07E-4)**	5.42E-2(2.57E-3)
	WFG7	**2.01E-3(3.71E-4)**	4.91E-2(4.73E-4)
3	DTLZ1	**6.69E-3(5.11E-4)**	1.09E-2(1.26E-2)
	DTLZ2	9.66E-3(1.01E-3)	**9.03E-3(7.21E-5)**
	DTLZ3	**1.35E-2(1.1E-2)**	3.4E-2(1.11E-1)
	DTLZ4	**1.14E-2(3.23E-3)**	1.73E-2(1.16E-2)
	WFG6	**1E-2(8.04E-4)**	1.17E-2(4.38E-4)
	WFG7	1.3E-2(4.86E-4)	**1.06E-2(5.07E-4)**
5	DTLZ1	**1.08E-2(5.93E-4)**	3.18E-2(2.45E-3)
	DTLZ2	**1.71E-2(9.74E-4)**	5.36E-2(2.33E-6)
	DTLZ3	**1.98E-2(3.53E-3)**	5.29E-2(1.47E-3)
	DTLZ4	**1.68E-2(1.6E-3)**	5.33E-2(2.22E-3)
	WFG6	**1.55E-2(8.82E-4)**	5.2E-2(7.53E-4)
	WFG7	**1.83E-2(8.68E-4)**	6.56E-2(4.09E-4)
8	DTLZ1	**2.47E-2(1.44E-3)**	4.38E-2(4.19E-4)
	DTLZ2	**5.08E-2(3.72E-3)**	7.38E-2(1.35E-6)
	DTLZ3	**5.35E-2(5.13E-3)**	7.38E-2(9.16E-5)
	DTLZ4	**3.84E-2(2.53E-3)**	7.32E-2(2.73E-3)
	WFG6	**4.36E-2(3.67E-3)**	7.7E-2(1.86E-3)
	WFG7	**4.75E-2(3.98E-3)**	2.12E-1(1.91E-2)
10	DTLZ1	**1.93E-2(9.84E-4)**	3.28E-2(7.38E-4)
	DTLZ2	**4.46E-2(1.8E-3)**	5.88E-2(9.75E-7)
	DTLZ3	**4.45E-2(4.29E-3)**	5.87E-2(1.34E-4)
	DTLZ4	**2.97E-2(2.2E-3)**	5.67E-2(2.52E-3)
	WFG6	**3.82E-2(2.73E-3)**	8E-2(1.07E-3)
	WFG7	**6.46E-2(4.88E-3)**	4.49E-1(5.79E-2)



Friedman test p-value: $p = 2.065286e - 06$

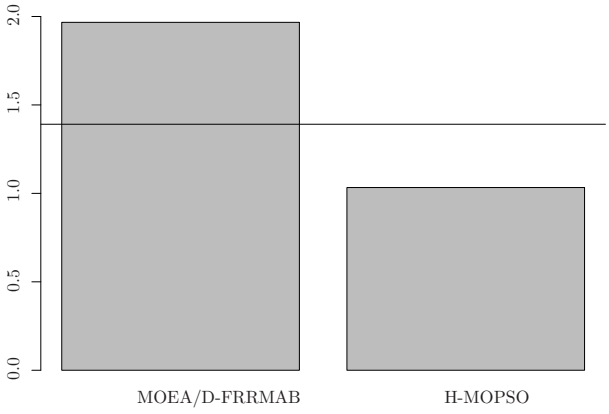
Figure 4.18: Average ranking H-MOPSO-ROULETTE and MOEA/D-FRRMAB for the IGD indicator

In the general analysis, considering all problems and objective numbers, presented in Figure 4.18, H-MOPSO-ROULETTE outperformed MOEA/D-FRRMAB again with a statistical difference according to the Friedman test.

By looking at the Hypervolume results, presented in Table 4.14, we can see that the H-MOPSO-ROULETTE can be considered clearly superior to MOEA/D-FRRMAB. The overall results presented in Figure 4.19 confirm this outcome, where H-MOPSO-ROULETTE outperforms MOEA/D-FRRMAB with a statistical difference according to the Friedman test.

Table 4.14: HV: Mean (and standard deviation) for H-MOPSO-ROULETTE and MOEA/D-FRRMAB

Obj.	Problem	ROULETTE	MOEA/D-FRRMAB
2	DTLZ1	**9.98E-1(4.52E-3)**	9.62E-1(1.34E-3)
	DTLZ2	**2.1E-1(4.38E-4)**	8.58E-2(2.27E-7)
	DTLZ3	**9.96E-1(1.17E-2)**	9.46E-1(1.78E-5)
	DTLZ4	**2.1E-1(2.16E-4)**	8.58E-2(1.94E-7)
	WFG6	**2.18E-1(7.57E-4)**	3.37E-2(1.92E-2)
	WFG7	**2.15E-1(5.01E-3)**	7.88E-2(4.81E-3)
3	DTLZ1	**9.75E-1(7.55E-4)**	9.61E-1(4.85E-2)
	DTLZ2	**3.98E-1(7.37E-3)**	3.91E-1(3.2E-3)
	DTLZ3	**1E0(1.31E-4)**	1E0(8.85E-4)
	DTLZ4	**4.19E-1(7.44E-3)**	3.83E-1(3.48E-2)
	WFG6	**3.93E-1(7.87E-3)**	3.32E-1(1.22E-2)
	WFG7	3.72E-1(1E-2)	**4.14E-1(1.19E-2)**
5	DTLZ1	**1E0(1.67E-9)**	9.71E-1(1.14E-2)
	DTLZ2	**8.73E-1(9.71E-3)**	1.07E-1(3.16E-5)
	DTLZ3	**1E0(1.23E-9)**	9.87E-1(8.98E-4)
	DTLZ4	**9.82E-1(1.35E-3)**	2.06E-1(4.64E-2)
	WFG6	**5.8E-1(1.44E-2)**	4.73E-2(5.65E-3)
	WFG7	**5.4E-1(1.79E-2)**	1.03E-1(1.42E-3)
8	DTLZ1	**1E0(4.82E-6)**	8.99E-1(6.25E-3)
	DTLZ2	**9.45E-1(1.21E-2)**	9.39E-2(2.12E-5)
	DTLZ3	**1E0(2.55E-9)**	9.87E-1(6.31E-5)
	DTLZ4	**1E0(1.52E-4)**	3.37E-1(7.58E-2)
	WFG6	**5.58E-1(4.97E-2)**	3.7E-2(3.46E-3)
	WFG7	**4.21E-1(2.84E-2)**	6.75E-2(1.5E-2)
10	DTLZ1	**1E0(1.74E-6)**	9.4E-1(2.45E-2)
	DTLZ2	**9.87E-1(7.86E-3)**	1.76E-2(8.75E-6)
	DTLZ3	**1E0(6.39E-11)**	9.77E-1(1.31E-3)
	DTLZ4	**1E0(2.62E-5)**	4.56E-1(1.43E-1)
	WFG6	**6.61E-1(5.66E-2)**	3.31E-2(2.01E-3)
	WFG7	**4.55E-1(3.48E-2)**	7.86E-2(2.02E-2)



Friedman test p-value: $p = 3.186355e-07$

Figure 4.19: Average ranking H-MOPSO-ROULETTE and MOEA/D-FRRMAB for the HV indicator

4.3 Discussion

This work presented H-MOPSO, a new MOPSO algorithm based on hyper-heuristic to select good leader and archiving methods. H-MOPSO can use any selection hyper-heuristic and is guided by the $R2$ performance indicator, which is a fast indicator that evaluates desired aspects of a Pareto front approximation. There are few works in the literature that apply hyper-heuristics on multi-objective optimizers, and this work, as far as we know, is the first using a hyper-heuristic to select leader and archiving methods in a MOPSO.

Several experimental studies were conducted to investigate the behavior of H-MOPSO on different scenarios, as well as to compare it with state-of-the-art algorithms:

In the first study, our goal is to assess if the H-MOPSO framework is able to guide the search to through the use of its hyper-heuristic. To achieve this goal, we compared the H-MOPSO using the simple roulette wheel hyper-heuristic with all its low-level heuristics used separately. This comparison was made using the $R2$ indicator, the same used within H-MOPSO with the goal of not evaluating the performance of the algorithm, but its capacity of guiding the search through the low-level heuristics. The results from this first study indicate that the proposed hyper-heuristic is robust and able to correctly guide the search by presenting good results in most of the cases.

In a second experimental study, we compared the performance of H-MOPSO using four heuristic selection methods, two state-of-the-art from the literature: Adaptive Choice Function (ACF) (Drake et al., 2012; Gonçalves et al., 2015b) and Fitness-Rate-Rank-based Multi-Armed Bandit (FRRMAB) (Gonçalves et al., 2015a; Li et al., 2014). The remaining two heuristic selection methods were the simple roulette wheel previously used and a random hyper-heuristic that employs no learning. From the results obtained in this study, it is possible to observe that although few significant statistical differences were found, in general, Roulette is able to achieve better average results than the other heuristic selection methods in most cases. This means that by using advanced heuristic selection methods, we are not able to achieve a significant improvement in the quality of the generated solutions.

In a third study, two characteristics of the hyper-heuristic are configured: the set of low-level heuristics to be dynamically replaced (i.e., only archiving, only leader selection or both methods simultaneously), and the interval used to replace the low-level heuristics. Our goal here is to identify if changing only a subset of low-level heuristics is best than changing all. It means if selecting an appropriate option for one method (archiving or leader selection) is better than selecting the proper options for both together. Moreover, we want to calibrate the interval used to replace the low-level heuristics to give them more time (in iterations number) to influence the search, hence allowing to evaluate if subsequent applications of the same hyper-heuristic can intensify its performance gain with the same overall computational budget. The results of this study show that, in general, lower values produce better results than higher, and the best value (with no significant differences) is changing at every iteration, like in the previous studies.

At a final study, we compare the H-MOPSO framework with the state-of-the-art MOEA/D-FRRMAB (Li et al., 2014) hyper-heuristic framework. For this experiment, we used the best parameter values and heuristic selection method found in the previous experiments. The results of these experiments indicate that the H-MOPSO framework can achieve excellent results, and even outperform a state-of-the-art algorithm in most of the investigated problems.

Despite our good results obtained so far, other directions of investigation can still be pursued for future works, like selecting more advanced move acceptance criteria for the hyper-heuristic or investigating alternative methods to assess the quality of the solutions obtained by the low-level heuristics.

Chapter 5

Investigating clustering strategies on IMulti

In multi-objective optimization, the two most important requirements of a generated Pareto set are its closeness to the true Pareto front and the diversity of solutions along it. However, with an increase in the number of objectives, the size of the objective space and the surface of the Pareto front can greatly increase, making much harder for an optimizer to achieve these requirements.

A possible alternative to deal with such problems is to use multiple populations (or swarms). These populations should be well spread over the entire search space in order to increase the diversity of solutions. Moreover, when each single population concentrates in a small portion of the search space, the individuals are able to specialize, leading to better convergence.

I-Multi (Britto et al., 2013) is a recently introduced MOPSO designed to deal with many-objective problems. A distinguished characteristic of I-Multi is that it uses multiple swarms to cover different areas of the objective space. Its search procedure can be divided in two phases: diversity and multi-swarm searches.

In its multi-swarm phase, I-Multi uses a clustering algorithm, so the repository of each sub-population is initialized with the solutions clustered together. Since these solutions are used to guide the search, this search concentrates in a smaller part of the search space. We argue that the clustering strategy employed in the multi-swarm phase of I-Multi, influences its search ability. We investigate this issue in this chapter, showing that the specific choice of the clustering strategy used by I-Multi can have an important impact in the behavior of the algorithm and in the quality of the Pareto fronts generated by the MOPSOs.

We focus on finding answers for the following questions: 1) What is the most efficient clustering strategy for MaOPs: clustering in the space of decision variables, objectives, or both? 2) Is it possible to characterize the type of problems (benchmark functions) for which one type of clustering is better than the others? 3) What is the influence of the similarity metric used for clustering in the behavior of the algorithms? 4) Among the metrics compared, which one contributes the most to obtaining good solutions? i.e. which is the best metric among those compared?

After conducting extensive experiments using two families of difficult benchmark functions of up to 20 objectives, we clearly identified clustering in the objective space as the most efficient clustering strategy in most of the cases. However, we also identified functions for which clustering in the decision space leads to better approximations of the Pareto front. Further examination of these cases allowed us to identify two modalities of difficulty that are particularly suited to be treated using decision space clustering. These are: bias and deception. Our study of the clustering metrics influence reveals a clearer scenario in which the use of other metrics, different to the commonly applied Euclidean distance, does not produce significant improvements in the general case. As far as we know, this is the first work where the performance of different clustering strategies for MOPs is linked to the modalities of difficulty of the functions. Similarly, we have not found any previous study on the impact of the similarity metrics on the behavior of the algorithms.

The remainder of this chapter is organized as follows: The I-Multi algorithm used here as a case study is explained in Section 5.1. Section 5.2 describes the investigated clustering strategies, and Section 5.3 presents the experimental study conducted to identify the best of these strategies. Finally, Section 5.4 presents a discussion of this chapter.

5.1 I-Multi

Recently a new MOPSO called Iterated Multi-swarm (I-Multi) was proposed by Britto et al. (2013). To enhance its performance it uses multiple swarms to cover different areas of the

search space. Each sub-swarm is defined by limiting its search around a point in the search space, therefore each sub-swarm will focus the search on smaller regions.

I-Multi execution can be divided into two phases: *diversity* and *multi-swarm* searches. In the first phase, a diversity search is executed for a predefined number of iterations, using SMPSO (Section 3.3.3) with the MGA archiver (Section 3.3.1.2) and its default CD archiver to generate a set of well-distributed non-dominated solutions (basis front F_b) for multi-swarm initialization.

Multi-swarm phase is designed to introduce convergence towards the Pareto front. This phase begins by using the K-Means algorithm (Hartigan and Wong, 1979) for clustering the solutions contained in the basis front (F_b) to generate a predefined number of sub-swarms (NS). The solutions from each cluster compose the initial repository of each sub-swarm (F_k) and the centroid of the cluster can be used as seed (S_k) for the swarm. Around each seed (within a specified search region ($\mathcal{V}$)), a set of solutions is randomly generated as particles of the sub-swarm. For a predefined number of iterations, each sub-swarm runs independently, using the SMPSO with the Ideal archiver (Britto and Pozo, 2012) and the same archiver as before to enhance its convergence. After that, the repository of each sub-swarm is integrated to the basis front, so only the non-dominated solutions regarding all repositories are kept. At the end of this process, the basis front is split into sub-swarms as before. This process of joining and splitting the fronts is called split iteration, and it is repeated a predefined number of times (SI). This process enables an indirect communication between the sub-swarms.

The process of a split iteration is depicted in Figure 5.1, where at first there is a single swarm whose solutions in the repository are represented as black circles and the particles are presented as white circles. Next, the repository (basis front) of the single swarm is split into a predefined number of clusters, where the solutions clustered together in each cluster becomes the initial repository of a sub-swarm and the centroid of this cluster can be used as seed for this sub-swarm. To complete each sub-swarm, a set of particles is randomly generated around the seed. After a predefined number of runs, all the non-dominated solutions regarding all clusters are combined again to form a new basis front and start a new split iteration.

The first step of a partition iteration is to define a seed to each sub-swarm, the seed represents the center of the search region of a sub-swarm. In (Britto et al., 2013) three methods to define the seed of the sub-swarm are presented: I-Multi Centroid, I-Multi Extremes and I-Multi Random.

In I-Multi Centroid, the centroid of each cluster is selected as seed for each sub-swarm and the solutions clustered around each seed will compose the repository of each sub-swarm. Ideally, by doing this clustering, the solutions will be clustered together both in the objective and decision spaces, but there is no guarantee for that, since the solutions were clustered only in the decision space.

In I-Multi Extremes, extreme solutions in terms of objective values are selected as seeds for the sub-swarms. Extreme solutions in terms of an objective function are solutions that present the best value for that objective. The repositories of the sub-swarms are filled with the solutions using the same policy. If the number of sub-swarms is greater than the number of objectives, the remaining seeds are selected randomly from the basis front to form a sub-swarm. If the opposite occurs, some dimensions would not be explored by the algorithm. The number of chosen solutions is equal to the archive size.

In I-Multi Random, all the seeds are randomly selected from the basis front. The repositories of the sub-swarms are also randomly filled and the number of chosen solutions is equal to the archive size.

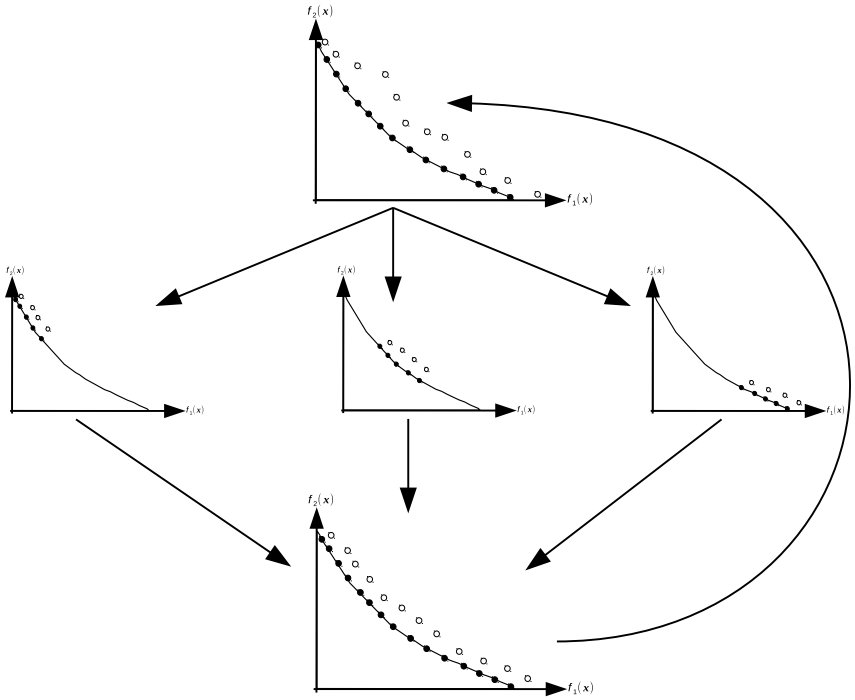


Figure 5.1: Representation of the I-Multi algorithm.

After the definition of the seeds, the search region of each sub-swarm must be defined. In this case, given the decision variable vector of the seed, for each dimension of this vector the value of the interval that defines the new search region is added (greatest value) and subtracted (smallest value). In I-Multi, this interval can be updated at each partitioning iteration by the following strategy: initial i_i and final i_f interval values are set by parameters. Next, a value is calculated to lead the interval value from the initial to the final along the partitioning iterations SI, this value is calculated through:

$$\mathcal{V} = \frac{|i_i - i_f|}{SI} \tag{5.1}$$

After de definition of the seeds and the intervals, each sub-swarm must be initialized, in this stage, the population is randomly initialized within the search region, next SMPSO using the Ideal archiver and CD leader is executed in each sub-region. Finally, at the end of the execution, all the non-dominated solutions found by each sub-swarm are merged into a new basis front. This process is repeated by a pre-defined number of partitioning iterations.

In a new iteration, new seeds are defined to the sub-swarms. Since the explored space is modified by the selection of a new seed, the populations of the swarms are reinitialized within the new limits. However the non-dominated solutions found by the sub-swarms are kept during the search.

In this algorithm the sub-swarms communicate in an indirect way. First, a sub-swarm can choose as seed a solution from the base front that has been found by other sub-swarm, since all the sub-swarms are re-defined. Also, at each new definition of seeds, the repository is updated with solutions from the base front that are similar to the seed, so a sub-swarm can add solutions from other sub-swarm to its repository.

Algorithm 10: I-Multi

```
   // Phase1: Diversity search
1  F_b=Run-MGA-SMPSO()
   // Phase 2: Multi-swarm search
2  for s=1 to SI do
3  |  F=SplitFront(F_b)
4  |  S=SelectSeed(F_b)
5  |  V=calculateInterval()
6  |  for k=1 to NS do
7  |  |  P_k=initializePop(S_k)
8  |  |  F_k=Run-Ideal-SMPSO(P_k,F_k)
9  |  end
10 |  F_b=Non-dominated(F)
11 end
12 return F_b
```

Algorithm 10 presents the pseudocode of I-Multi. Firstly SMPSO is run using the MGA archiver and CD leader to generate the basis front. Then, the basis front is split into NS fronts and the seeds of each sub-swarm are defined, also the search interval for this split iteration is calculated.

For each sub-swarm, a population P_k is randomly initialized within the search region by using the seed S_k and the search interval. Next the repository is updated with the solutions found by the sub-swarm running SMPSO with Ideal archiver. After the execution of all the sub-swarms, the non-dominated solutions regarding all the repositories replace the basis front and the split iteration repeats for a predefined number of times. At the end of the search, the last basis front is returned as result of the search.

5.2 Clustering strategies for multi-objective problems

In this section we present the different clustering strategies that have been implemented as part of I-Multi, explaining the rationale behind their choice.

5.2.1 Components of the clustering algorithms

The following elements influence the behavior of the clustering strategies when used within many-objective optimizers:

1. Clustering space explored: decision space, objective space or a combination of both spaces.
2. Similarity measure employed to compute the clusters.
3. Number of clusters.
4. Type of clustering algorithm.

In this paper we focus on the first two strategies. As illustrated in the review of related work, clustering in the decision and objective spaces is extensively applied, but the impact of the choice on the behavior of the EA is usually not addressed. Similarly, the issue of the similarity

measure applied is commonly overlooked. The influence of the number of clusters has been investigated in previous works (Britto et al., 2013; Castro Jr. et al., 2016b) and the effect that the type of clustering algorithm may have in the search is left for future work.

The space where the solutions are clustered can play an important role in the optimization problems. Its effect can be even greater depending on whether or not points nearby in objective space are also close in the decision space.

In the original I-Multi paper (Britto et al., 2013), the objective of clustering in the decision space is mostly to increase the convergence of the algorithm by means of specialization of the solutions. Since each cluster has very similar solutions, the interactions among their decision vectors produce small perturbations, and consequently the exploitation of a small part of the decision space is increased.

Here we propose to change the space to which clustering is applied from the decision to the objective space. By making this change, we expect to achieve more diversity in both spaces, since each cluster will be concentrated in a different region of the Pareto front. However, the decision variables of the solutions within the same cluster will not necessarily take similar values, hence the interactions among these solutions are more likely to generate greater perturbations, leading to the exploration of a larger part of the search space. By combining the decision and objective spaces of the solutions to conduct the clustering, both components make a contribution to the computation of the similarity measure during clustering. Consequently a good trade-off between convergence and diversity is expected.

The similarity between two solutions can be interpreted differently depending on the space used to measure it. However, a different interpretation of this similarity can also be obtained depending on the indicator used to measure this similarity. By changing the similarity metric, the shape and location of the clusters are changed as well, and consequently the metric used has an impact on the behavior of the search algorithm.

5.2.2 Clustering space

In this work we investigate two different alternative spaces to perform the clustering procedure in I-Multi: clustering in the objective space and in an alternative space that we called *both*, and is composed of a combination of objective and decision spaces.

In the traditional (decision space clustering) approach used by I-Multi, the centroids of the clusters are used as "seeds", i.e., solutions around which the search region is defined. However, when doing clustering in the objective space, we cannot use the centroids of the clusters found by K-Means as seeds. Instead, in this approach we set each position of the seed as the average of each position of the decision variables of the solutions whose objective vectors have been grouped in the cluster.

Our other proposed approach (both) uses a combination of both spaces in order to cluster the solutions. In this case, the K-Means algorithm is executed in the space defined by the concatenation of decision variables and objectives $\boldsymbol{c} = (u_1, ..., u_m, x_1, ..., x_n)$, where $\boldsymbol{u} = \boldsymbol{f}(\boldsymbol{x})$. m is the number of objectives and n is the number of decision variables. Since this approach uses the decision variables as part of the clustering space, we use the last n elements of the centroid found by K-Means as the swarm seed.

5.2.3 Measures of similarity

The metric used to evaluate the similarity between the solutions has an influence on the results of the clustering algorithm. The clustering metrics define different ways to look at the

similarity relationships between the solutions. Previous works in the areas of machine learning and pattern recognition (Aggarwal et al., 2001; Deborah et al., 2015; Howarth and Rüger, 2005) have acknowledged the impact that the choice of the distance metric has on different algorithms when computing the similarity between solutions is required. For instance, the so called *fractional distance* metrics have shown to significantly improve the effectiveness of clustering algorithms for high dimensional problems (Aggarwal et al., 2001). As part of our study, we have selected a set of representative similarity metrics that include the Euclidean distance (Deza, 2009), the most commonly applied distance metric in optimization algorithms, and a set of other metrics extensively applied in other areas but rarely investigated in the context of optimization algorithms. One of the implicit questions we address is whether clustering algorithms that use such metrics can promote better results than those that employ the Euclidean metric. Therefore, as part of our study, we investigated a set of representative similarity metrics and the sensitivity of I-Multi to this choice.

The Minkowski distance (Deza, 2009) between two vectors $\boldsymbol{x} = (x_1, ..., x_n)$ and $\boldsymbol{y} = (y_1, ..., y_n)$ is defined as:

$$\left(\sum_{i=1}^{n} \mid x_i - y_i \mid^{M_k} \right)^{\frac{1}{M_k}} \tag{5.2}$$

where M_k is a parameter of the Minkowski metric that defines a family of metrics. The effect of changing the value of M_k is displayed in Figure 5.2. In our experiments, we considered $M_k \in \{0.5, 2, 4, \infty\}$ as parameters of the Minkowski metric because these values represent the most commonly applied distance metric (Euclidean) and a number of other metrics extensively applied in other areas. Moreover, these values allow us to explore a variety of scenarios in terms of how to compute the similarity between the solutions, and in particular the weight given to the difference in the components of the vectors.

For $M_k = 2$, the Euclidean distance is obtained from Equation (5.2), and for $M_k = \infty$ we obtain the Tchebycheff (Deza, 2009) distance. For $M_k = 0.5$, and, in general, for $M_k \in (0, 1)$ these measures, usually called fractional distance metrics (Aggarwal et al., 2001), are not metrics since the triangle inequality is violated. However, they still convey a sense of closeness and have been shown to produce excellent results in practice (Deborah et al., 2015; Howarth and Rüger, 2005). For $M_k = 0.5$, the Minkowski distance exhibits properties that are midway between the properties of the Euclidean distance ($M_k = 2$) and ($M_k = 0$), a fact that makes it worth of investigation.

5.3 Empirical study

The primary objectives of the experiments are the following: Q1) To find out which of the clustering strategies produces the best results when used within I-Multi. Q2) To determine the impact that the choice of the distance metric has on the results of I-Multi. Q3) To identify or unveil any type of casual relationship between the characteristics of the optimization problem (number of objectives, deception, multi-modality or bias) and the behavior of I-Multi when the different clustering strategies are applied.

Finding an answer to question Q1 will contribute to a better understanding of the behavior of the I-Multi algorithm and MOPSOs in general. Similarly, investigating question Q2 will help to ascertain if the popular assumption of using the Euclidean distance between solutions is the right choice, or if the results of MOPSOs that apply clustering techniques could be further improved by using other metrics.

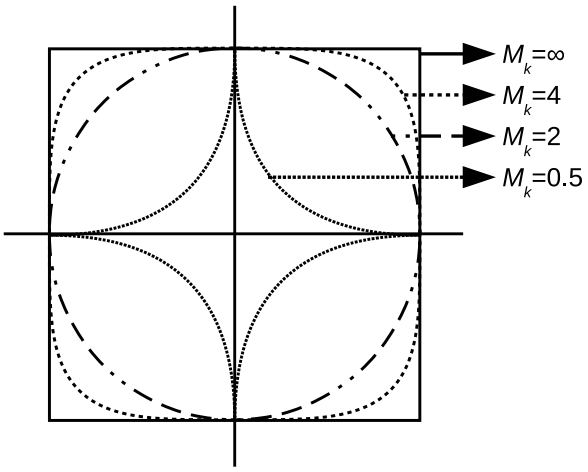


Figure 5.2: Unit circles of the Minkowski metric with various values of M_k.

We will empirically address questions Q1 and Q2 by employing the I-Multi algorithm using different methods on a representative set of difficult multi-objective functions for which a characterization of their domains of difficulty exists. We will evaluate the quality of the Pareto fronts obtained using the different clustering strategies.

Question Q3 helps us to get some insight about how the characteristics of the functions being optimized make the application of the different clustering strategies particularly suitable for each characteristic. This potential mapping between the characteristics of the functions and the "most promising" strategy for each characteristic is important, since it allows the user to have at least some heuristic criteria to decide in which situations a clustering strategy is expected to behave better than the others. We address question Q3 by first detecting characteristic patterns of algorithm behavior for each of the functions, and conceiving additional experiments to test alternative hypotheses that explain this behavior from the characteristics of the functions.

5.3.1 Experimental setup

The parameters of the algorithms used in our experiments are summarized in Table 5.1.

Table 5.1: Parameters of I-Multi

C_1, C_2	varies randomly in [1.5,2.5]
Objectives (m)	3, 5, 8, 10, 15 and 20
Initial phase duration	100 iterations
Initial phase population	100 particles
Multi-swarm phase duration	100 iterations
Number of swarms & clusters	50
Multi-swarm phase population	(750/number of swarms) particles
Repository maximum size	200 solutions
Multi-swarm region size	decrease from 0.5 to 0.1
Split iterations	5

The parameters C_1 and C_2 control the effect of the personal and global best particles in the velocity calculation, respectively. They are set according to the recommendation given in the original SMPSO paper (Nebro et al., 2009). The number of decision variables was set according to the recommendation given for the problems by Deb et al. (2005) and Huband et al. (2006). The number of split iterations, as well as the multi-swarm region size were calibrated by Britto et al. (2013) and we use the best values found. The number of iterations (initial and total), the initial size of the population, the maximum size of the repositories, and the total number of

particles were also set as proposed by Britto et al. (2013). The number of swarms and clusters were set according to previous work (Castro Jr. et al., 2016b) with a similar variant of I-Multi. Regarding the number of particles per swarm, if the total number is not divisible by the number of sub-swarms, the remaining particles are distributed among the sub-swarms. As in (Castro Jr. et al., 2016b), we used the crowding distance (CD) (Deb et al., 2000) archiver in the multi-swarm phase.

5.3.2 Comparison between the spaces of clustering

This section presents the results of the different strategies used by I-Multi to cluster the solutions. Each of these strategies is defined by the space in which clustering is accomplished.

5.3.2.1 DTLZ benchmark

Table 5.2: Mean ranks of the IGD_p as used in the Kruskal-Wallis test for the DTLZ problems. Final ranks, presented in parentheses assigned according to mean ranks.

Obj.	Spaces	DTLZ1	DTLZ2	DTLZ3	DTLZ4	DTLZ5	DTLZ6	DTLZ7
3	Objective	**51.33 (2.00)**	57.87 (2.50)	**42.87 (2.00)**	**45.87 (2.00)**	74.30 (3.00)	**19.73 (1.00)**	**40.53 (2.00)**
	Both	**47.13 (2.00)**	56.33 (2.50)	**50.77 (2.00)**	**47.60 (2.00)**	**34.60 (1.50)**	56.00 (2.50)	**46.30 (2.00)**
	Decision	**38.03 (2.00)**	**22.30 (1.00)**	**42.87 (2.00)**	**43.03 (2.00)**	**27.60 (1.50)**	60.77 (2.50)	**49.67 (2.00)**
5	Objective	**51.40 (2.00)**	**20.40 (1.00)**	**42.53 (2.00)**	54.07 (2.50)	**29.50 (1.00)**	**28.43 (1.00)**	**46.23 (2.00)**
	Both	**46.43 (2.00)**	46.63 (2.00)	**49.27 (2.00)**	46.07 (2.00)	45.83 (2.50)	58.43 (2.50)	**48.17 (2.00)**
	Decision	**38.67 (2.00)**	69.47 (3.00)	**44.70 (2.00)**	**36.37 (1.50)**	61.17 (2.50)	49.63 (2.50)	**42.10 (2.00)**
8	Objective	**50.63 (2.00)**	**15.50 (1.00)**	**42.70 (2.00)**	62.67 (3.00)	**21.30 (1.00)**	**34.13 (1.50)**	**42.80 (2.00)**
	Both	**46.03 (2.00)**	45.50 (2.00)	**48.00 (2.00)**	**42.10 (1.50)**	54.87 (2.50)	49.97 (2.00)	**48.70 (2.00)**
	Decision	**39.83 (2.00)**	75.50 (3.00)	**45.80 (2.00)**	**31.73 (1.50)**	60.33 (2.50)	52.40 (2.50)	**45.00 (2.00)**
10	Objective	54.57 (2.50)	**15.50 (1.00)**	**39.43 (2.00)**	69.07 (3.00)	**19.10 (1.00)**	**34.03 (1.50)**	**45.50 (2.00)**
	Both	47.90 (2.00)	47.53 (2.00)	**45.93 (2.00)**	**41.43 (1.50)**	53.17 (2.50)	56.82 (2.50)	**42.27 (2.00)**
	Decision	**34.03 (1.50)**	73.47 (3.00)	**51.13 (2.00)**	**26.00 (1.50)**	64.23 (2.50)	45.65 (2.00)	**48.73 (2.00)**
15	Objective	58.53 (2.50)	**15.50 (1.00)**	**49.80 (2.00)**	75.00 (3.00)	**21.27 (1.00)**	**36.70 (2.00)**	**30.13 (1.00)**
	Both	44.33 (2.00)	56.53 (2.50)	**42.70 (2.00)**	**38.50 (1.50)**	57.07 (2.50)	**50.83 (2.00)**	55.40 (2.50)
	Decision	**33.63 (1.50)**	64.47 (2.50)	**44.00 (2.00)**	**23.00 (1.50)**	58.17 (2.50)	**48.97 (2.00)**	50.97 (2.50)
20	Objective	**46.43 (2.00)**	**15.50 (1.00)**	**43.60 (2.00)**	75.47 (3.00)	**17.33 (1.00)**	**31.80 (1.00)**	45.17 (2.00)
	Both	**44.93 (2.00)**	56.03 (2.50)	**49.57 (2.00)**	**38.33 (1.50)**	57.70 (2.50)	53.87 (2.50)	54.47 (2.50)
	Decision	**45.13 (2.00)**	64.97 (2.50)	**43.33 (2.00)**	**22.70 (1.50)**	61.47 (2.50)	50.83 (2.50)	**36.87 (1.50)**

Table 5.3: Overall ranks of the IGD_p as used in the Friedman test for all the DTLZ problems and numbers of objectives. Final ranks, presented in parentheses assigned according to the ranks.

Objectives	Both	Decision
75.0 (2.0)	**96.0 (2.0)**	**81.0 (2.0)**

Table 5.4: Overall ranks of the Hypervolume as used in the Friedman test for all the DTLZ problems and numbers of objectives. Final ranks, presented in parentheses assigned according to the ranks.

Objectives	Both	Decision
71.5 (2.0)	**92.5 (2.0)**	**88.0 (2.0)**

Table 5.2 shows the IGD_p results. In this table, the first column represents the number of objectives, the second column represents the different strategies: *Objective* indicates that clustering is made in the objective space. *Both* indicates that the clustering is made in a combination of both, the objectives and decision spaces. *Decision* means that the clustering is made in the decision space. The other columns show the mean ranks of the results obtained using each clustering space.

From the analysis of Table 5.2, we can group the behavior of the three algorithms on all the functions into three classes. 2) No statistical differences between the three algorithms across different number of objectives (Functions DTLZ3 and to a lesser extent functions DTLZ1 and DTLZ7). 2) Statistical differences indicate that objective space clustering is the best choice (functions DTLZ2, DTLZ5, and DTLZ6). 3) Statistical differences indicate that objective space clustering is the worst (function DTLZ4), while decision space clustering is the best.

In this classification, we have extracted global behavior patterns regarding the clustering space according to specific problems and objective numbers. We considered the general rankings as a secondary measure of difference between the algorithms, even if statistical differences were not found. A summarized analysis, considering all the combinations of problems and numbers of objectives (seven problems and six numbers of objectives, on a total of 42 subproblems) for each algorithm is presented in Table 5.3 for the IGD_p indicator and in Table 5.4 for the Hypervolume indicator.

In these tables we can see that the summarized analysis erases the individual differences detected for each function. However, it can be appreciated that clustering in the objective space has a lower value for the average ranking, i.e., its global results (considering the 42 subproblems) are slightly better than when considering both indicators.

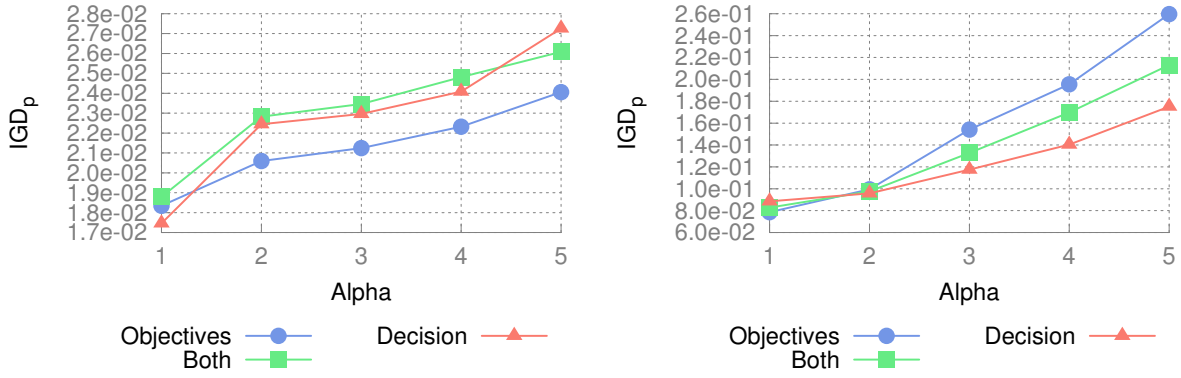
In the next step, we focus on unveiling the characteristics of the functions that influence the behavior of the clustering strategies. To do so, we present a comparative analysis of the behavior of the algorithms on functions DTLZ2 and DTLZ4. These functions are particularly interesting, because despite of presenting similar expression (as shown in Equations (5.3) and (5.4)), they are exemplars of classes 2 and 3 previously described, i.e., the clustering strategies present an opposite behavior when optimizing these functions.

$$
\begin{aligned}
Min \quad & f_1(\boldsymbol{x}) = (1 + g(x_m))cos(x_1\tfrac{\pi}{2}) \cdots cos(x_{m-1}\tfrac{\pi}{2}), \\
Min \quad & f_2(\boldsymbol{x}) = (1 + g(x_m))cos(x_1\tfrac{\pi}{2}) \cdots sin(x_{m-1}\tfrac{\pi}{2}), \\
\vdots \quad & \qquad \vdots \\
Min \quad & f_{m-1}(\boldsymbol{x}) = (1 + g(x_m))cos(x_1\tfrac{\pi}{2})sin(x_2\tfrac{\pi}{2}), \\
Min \quad & f_m(\boldsymbol{x}) = (1 + g(x_m))sin(x_1\tfrac{\pi}{2}), \\
& 0 \leq x_i \leq 1, \text{ for } i = 1, 2, ..., n, \\
where \quad & g(x_m) = \textstyle\sum_{x_i \in x_m}(x_i - 0.5)^2.
\end{aligned}
\tag{5.3}
$$

$$
\begin{aligned}
Min \quad & f_1(\boldsymbol{x}) = (1 + g(x_m))cos(x_1^{\alpha}\tfrac{\pi}{2}) \cdots cos(x_{m-1}^{\alpha}\tfrac{\pi}{2}), \\
Min \quad & f_2(\boldsymbol{x}) = (1 + g(x_m))cos(x_1^{\alpha}\tfrac{\pi}{2}) \cdots sin(x_{m-1}^{\alpha}\tfrac{\pi}{2}), \\
\vdots \quad & \qquad \vdots \\
Min \quad & f_{m-1}(\boldsymbol{x}) = (1 + g(x_m))cos(x_1^{\alpha}\tfrac{\pi}{2})sin(x_2^{\alpha}\tfrac{\pi}{2}), \\
Min \quad & f_m(\boldsymbol{x}) = (1 + g(x_m))sin(x_1^{\alpha}\tfrac{\pi}{2}), \\
& 0 \leq x_i \leq 1, \text{ for } i = 1, 2, ..., n, \\
where \quad & g(x_m) = \textstyle\sum_{x_i \in x_m}(x_i - 0.5)^2.
\end{aligned}
\tag{5.4}
$$

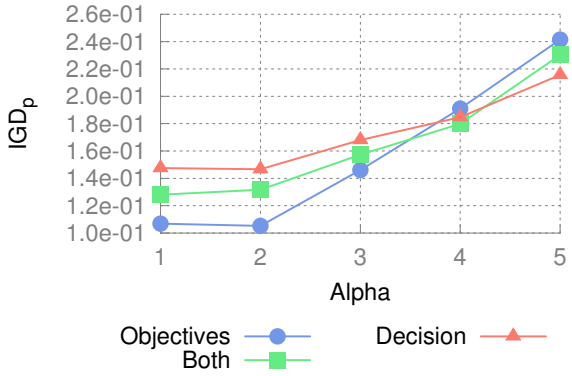
Since the only difference between the functions is the bias, represented by the parameter α in Equation (5.4), this source of difficulty seems to be the one that determines the opposite

behavior of the clustering strategies. We conducted a detailed analysis of the behavior of the clustering strategies for different values of α in Equation (5.4).



(a) 3 objectives

(b) 5 objectives



(c) 8 objectives

Figure 5.3: Function DTLZ4 with different α values for 3, 5 and 8 objectives. Average indicator results for the Pareto approximations obtained by I-Multi with different clustering strategies.

Figure 5.3 shows the average of the IGD_p results when optimizing the problem DTLZ4 for three, five and eight objectives with different values of α, remembering that smaller IGD_p values are better. As can be seen, along with the increase of the α value (consequently the bias), the results obtained using each clustering space become closer to those obtained using the DTLZ4 function and farther from those using the DTLZ2 function, where $\alpha = 1$ (refer to Table 5.2). Moreover, as the number of objectives increases, the sensibility of the algorithm to the increase in the bias seems to become smaller, where a change in the best clustering strategy requires a larger α value. Another interesting finding of the analysis is that increasing the bias worsens the quality of the Pareto approximations found by all variants of the algorithm.

5.3.2.2 WFG benchmark

Table 5.5 shows the results of the clustering strategies on the WFG benchmark. From the analysis of Table 5.5, it is possible to conclude that only classes 2 and 3 arise for this benchmark. Clearly, in most problems, it is preferable to cluster the solutions in the objective space (Class 2). The most notable exception is problem WFG5, in which clustering in the objective space is the worst for all numbers of objectives and, in general, it is better to cluster in both spaces (Class 3).

Table 5.5: Mean ranks of the IGD_p as used in the Kruskal-Wallis test for the WFG problems. Final ranks, presented in parentheses assigned according to mean ranks.

Obj.	Spaces	WFG1	WFG2	WFG3	WFG4	WFG5	WFG6	WFG7	WFG8	WFG9
3	Objective	**20.90 (1.00)**	**46.03 (2.00)**	**15.50 (1.00)**	**21.27 (1.00)**	75.50 (3.00)	75.20 (3.00)	**15.50 (1.00)**	**15.50 (1.00)**	75.47 (3.00)
	Both	52.87 (2.50)	**43.67 (2.00)**	51.37 (2.00)	41.47 (2.00)	**26.97 (1.50)**	**31.30 (1.50)**	50.13 (2.00)	50.40 (2.00)	**28.13 (1.50)**
	Decision	62.73 (2.50)	**46.80 (2.00)**	69.63 (3.00)	73.77 (3.00)	**34.03 (1.50)**	**30.00 (1.50)**	70.87 (3.00)	70.60 (3.00)	**32.90 (1.50)**
5	Objective	**16.93 (1.00)**	**25.93 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	75.50 (3.00)	75.03 (3.00)	**15.50 (1.00)**	**15.50 (1.00)**	**29.57 (1.00)**
	Both	59.40 (2.50)	52.57 (2.50)	54.50 (2.50)	68.23 (2.50)	**26.43 (1.50)**	**28.73 (1.50)**	53.10 (2.50)	53.83 (2.50)	50.27 (2.50)
	Decision	60.17 (2.50)	58.00 (2.50)	66.50 (2.50)	52.77 (2.50)	**34.57 (1.50)**	**32.73 (1.50)**	67.90 (2.50)	67.17 (2.50)	56.67 (2.50)
8	Objective	**34.60 (1.50)**	**15.70 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	75.50 (3.00)	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**16.73 (1.00)**
	Both	49.00 (2.00)	54.70 (2.50)	53.47 (2.50)	67.33 (2.50)	**22.17 (1.00)**	62.00 (2.50)	61.70 (2.50)	52.43 (2.00)	55.87 (2.50)
	Decision	52.90 (2.50)	66.10 (2.50)	67.53 (2.50)	53.67 (2.50)	38.83 (2.00)	59.00 (2.50)	59.30 (2.50)	68.57 (3.00)	63.90 (2.50)
10	Objective	**35.37 (1.50)**	**15.60 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	75.50 (3.00)	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**17.03 (1.00)**
	Both	53.87 (2.50)	51.10 (2.00)	56.33 (2.50)	71.53 (3.00)	**28.63 (1.50)**	62.60 (2.50)	59.97 (2.50)	54.27 (2.50)	57.83 (2.50)
	Decision	47.27 (2.00)	69.80 (3.00)	64.67 (2.50)	49.47 (2.00)	**32.37 (1.50)**	58.40 (2.50)	61.03 (2.50)	66.73 (2.50)	61.63 (2.50)
15	Objective	**51.47 (2.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	75.50 (3.00)	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**18.63 (1.00)**
	Both	**43.63 (2.00)**	52.10 (2.00)	56.80 (2.50)	72.03 (3.00)	**30.50 (1.50)**	61.63 (2.50)	58.53 (2.50)	52.57 (2.00)	52.53 (2.50)
	Decision	**41.40 (2.00)**	68.90 (3.00)	64.20 (2.50)	48.97 (2.00)	**30.50 (1.50)**	59.37 (2.50)	62.47 (2.50)	68.43 (3.00)	65.33 (2.50)
20	Objective	56.23 (2.50)	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	75.50 (3.00)	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**19.23 (1.00)**
	Both	**39.63 (1.50)**	58.70 (2.50)	54.57 (2.50)	69.70 (3.00)	**31.33 (1.50)**	63.53 (2.50)	56.27 (2.50)	55.13 (2.50)	52.50 (2.50)
	Decision	40.63 (2.00)	62.30 (2.50)	66.43 (2.50)	51.30 (2.00)	**29.67 (1.50)**	57.47 (2.50)	64.73 (2.50)	65.87 (2.50)	64.77 (2.50)

Table 5.6: Overall ranks of the IGD_p as used in the Friedman test for all the WFG problems and numbers of objectives. Final ranks, presented in parentheses assigned according to the ranks.

Objectives	Both	Decision
75.0 (1.0)	110.0 (2.0)	139.0 (3.0)

Table 5.7: Overall ranks of the Hypervolume as used in the Friedman test for all the WFG problems and numbers of objectives. Final ranks, presented in parentheses assigned according to the ranks.

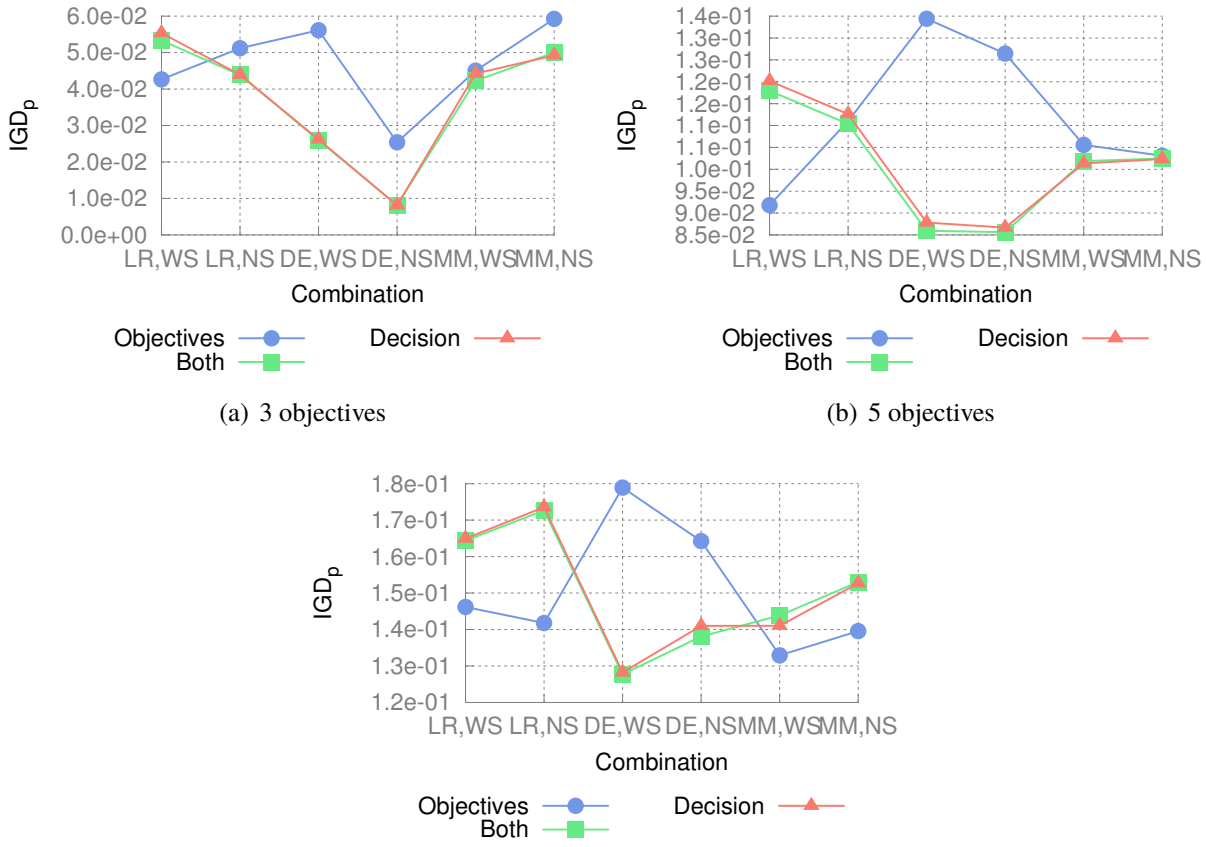
Objectives	Both	Decision
97.0 (1.5)	**97.0 (1.5)**	130.0 (3.0)

The differences presented in Table 5.5 are quite visible, however we made a global analysis of the results as in the previous benchmark. Tables 5.6 and 5.7 show the global analysis considering the IGD_p and Hypervolume indicators respectively. When considering the IGD_p it is more advantageous to cluster in the objective space, followed by both spaces, while according to the Hypervolume indicator clustering in the objective space or in both is equally good. IGD_p and Hypervolume indicate that clustering in the decision space, as done in (Britto et al., 2013), produces the worst results. Notice that there are significant statistical differences between the three strategies. Since the ranks of the results obtained with each space can be ordered as $objective < both < decision$, we can assume that the only reason why using the combined space (both) is better than the decision space is because it is composed of the objective space as well.

The fact that for most of the functions the objective clustering strategy is the best, makes the case of function WFG5 more intriguing. In addition to being the only function for which clustering the decision variables is better, WFG5 shares some of the characteristics of function WFG4. Therefore, we conducted a similar analysis to that presented in the previous section for functions DTLZ2 and DTLZ4. Our goal is to identify the characteristics of WFG5 that make

decision clustering a more efficient algorithm for this function. Function WFG5 defined in Equation 5.5, where $|\boldsymbol{z}| = n = p + d$ and $\boldsymbol{y} = \boldsymbol{z}_{[0,1]} = (\frac{z_1}{2}, ..., \frac{z_n}{2n})$.

$$
\begin{array}{llcl}
Given & \boldsymbol{z} & = & (z_1, ..., z_p, z_{p+1}, ..., z_n) \\
Min & f_1(\boldsymbol{x}) & = & x_m + 2\prod_{j=1}^{m-1} sin(x_j \frac{\pi}{2}) \\
 & f_{i=2:m-1}(\boldsymbol{x}) & = & x_m + 2i(\prod_{j=1}^{m-i} sin(x_j \frac{\pi}{2})) \cos(x_{m-i+1} \frac{\pi}{2}) \\
 & f_m(\boldsymbol{x}) & = & x_m + 2mcos(x_1 \frac{\pi}{2}) \\
Where & x_{i=1:m-1} & = & r\_sum((y_{\frac{(i-1)p}{(m-1)+1}}, ..., y_{\frac{ip}{(m-1)}}), (1, ..., 1)) \\
 & x_m & = & r\_sum((y_{p+1}, ..., y_n), (1, ..., 1)) \\
 & y_{i=1:n} & = & s\_deceptive(y_i, 0.35, 0.001, 0.05)
\end{array}
\tag{5.5}
$$



(a) 3 objectives

(b) 5 objectives

(c) 8 objectives

Figure 5.4: WFG functions with different combinations of transformations for 3, 5 and 8 objectives. Average indicator results for the Pareto approximations obtained by I-Multi with different clustering strategies.

Figure 5.4 shows the average of the IGD_p results when optimizing instances of three, five and eight objectives of WFG problems modified with different combinations of transformation functions. Here, we combine the three shift functions available (linear (LR), deceptive (DE) and multi-modal (MM)), with the two reduction functions (weighted sum (WS) and non-separable (NS)). The bias functions were not used because neither WFG4 nor WFG5 are biased but display different results, hence the bias is not determinant in this case.

We also want to highlight that some combinations tested are already part of the WFG family: WFG4 is multi-modal, weighted sum; WFG5 is deceptive, weighted sum; WFG6 is linear, non-separable. The combination linear, weighted sum can be considered an unbiased version of WFG7 and WFG8, since the only difference between these function is that if the bias is applied in the position-related or in the distance-related decision variables. Other combinations are found in part of the decision variables of WFG9: deceptive, non-separable (position-related), and multi-modal, non-separable (distance-related).

As can be seen from Figure 5.4, for the linear and multi-modal problems it is harder to recommend a clustering space, since the best space changes according to the objective number. On the other hand, in all the numbers of objectives studied, clustering in the objective space is the worst when the problem has a deceptive component, hence in these cases, clustering in decision space or in both spaces is recommended. This similar behavior when using both spaces or only the decision space can be explained by the fact that we have 24, 28 and 34 decision variables for only 3, 5 and 8 objectives respectively, since these number of variables for these problems and objective numbers are recommended by Huband et al. (2006). Therefore, the decision space has a larger weight in the similarity metric calculation and influences more than the objective space when using both spaces. From these results, we conclude that the deceptive character of the function has the main effect in the deterioration of the results of the objective clustering strategy. The relevant finding here is that clustering in the decision space is considerably less sensitive to this effect.

5.3.2.3 Correlation between quality of solutions and quality of clustering

In this section we will investigate the relationship between the quality of the clustering and the quality of the solutions obtained during the search. Quality of clustering is understood as how well similar solutions are grouped together, and how well these groups are separated. As a measure of clustering quality we use the Davies-Bouldin index (DB) (Davies and Bouldin, 1979). Similarly, we evaluate the quality of the solutions using the IGD_p metric because it is computationally cheap to compute. We then investigate the correlation between DB and IGD_p.

In order to calculate this correlation, we computed the DB and IGD_p at each iteration of the multi-swarm phase (last 100 iterations) of I-Multi using Euclidean distance as similarity metric. Then we averaged these 100 values over the 30 independent runs and used these averaged values to calculate the correlation. To compute the DB, first the scatter within each cluster was calculated by:

$$T_i = \left\{ \frac{1}{|F_i|} \sum_{j=1}^{|F_i|} |x_j - S_i|^q \right\}^{\frac{1}{q}} \tag{5.6}$$

where x_j is one point assigned to cluster F_i, S_i is the centroid of the cluster and $|F_i|$ is its size. Usually, the value of q is 1, so T_i becomes the average Euclidean distance between the points in the cluster and its centroid. Hence the separation between the clusters is measured as:

$$M_{ij} = |S_i - S_j|^p \tag{5.7}$$

Next, a measure of quality of each cluster is calculated by:

$$R_{ij} \equiv \frac{T_i + T_j}{M_{ij}} \tag{5.8}$$

and this is used to calculate R_i as $\max\{R_{ij_{i \neq j}}\}$. Finally, the DB index is calculated as:

$$DB \equiv \frac{1}{NS} \sum_{i=1}^{NS} R_i \tag{5.9}$$

where NS is the number of clusters.

DB is the system-wide average of the similarity measures of each cluster with its most similar cluster. The "best" choice of cluster will be that which minimizes this average similarity.

To investigate the relationship between IGD_p and DB, we selected four representative optimization functions: two problems from each set of benchmark functions and from these, we selected problems where the best results were achieved by clustering in the decision and objective spaces. These four problems can be considered representative for the entire set of benchmark functions used. The selected functions were DTLZ2, DTLZ4, WFG4 and WFG5.

The results obtained in this study are presented in Figure 5.5. This figure presents the correlation between the clustering quality and the Pareto fronts quality along the runs of the I-Multi variants using the four optimization problems with different dimensions (i.e., objective numbers).

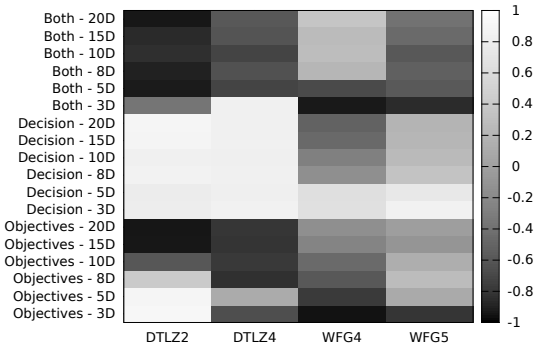


Figure 5.5: Heatmap of the correlation between the IGD_p and the DB.

In this figure, we can identify some patterns: the first is that in many cases, we have ascending or descending intensity of shades when considering the increase in the number of objectives. This behavior indicates that the increase in the number of objectives has an influence on the difficulty of the problem and/or clustering quality. A second identified pattern is that the increase or decrease of the correlation along the increase in the number of objectives occurs very differently according to the clustering space used. Furthermore, the differences between clustering in the objective space and in the decision space seem even greater, achieving an opposite behavior in some cases, such as in problems WFG4 and WFG5. This difference in behavior indicates that the clustering space used strongly influences the inner working of the algorithm.

The analysis of the results also reveals that, while for the clustering in the objective space and in both spaces the correlations are mostly negative, for clustering in the decision space

this correlation is strongly positive for three of the four functions, and close to zero in most cases for the fourth function.

5.3.2.4 Summary of the results

We summarize some of the main findings extracted from the experiments described in this section. In general, clustering in the objective space is recommended in both, the DTLZ and the WFG problems, since it achieves the best performance in most of the cases. However, this clustering space is more sensitive to properties of the problem, such as bias or deception, and can yield poor results in such conditions.

In the investigation of the correlation between the quality of the clustering and the quality of the solutions generated by the algorithm at each generation, we have seen that the space in which the clustering is done is an important factor to determine the strength of this correlation. Hence, it is expected that the choice of the clustering space affects the results obtained by the algorithm.

5.3.3 Comparison between distance metrics

In this section we investigate another significant question, i.e., whether and how the choice of the similarity metric influences the results of the clustering strategy. Building on the results shown in previous sections in which we found that, in general, better results can be obtained using the clustering in the objective space, we constrain the study of the metrics to I-Multi using this clustering space.

In the previous section, we used the K-Means algorithm. This algorithm was designed to work with the Euclidean distance and uses this metric in two places: explicitly in the distance calculation to allocate the closest points to a cluster and implicitly in the calculation of the new centroids that are defined as the mean of the points in a cluster. However, when we change the similarity metric employed, the mean may no longer represent the center of a cluster from the point of view of the new metric.

One alternative to solve this issue is to change the method used to define the center of the cluster. In this section we used the medoid as center, which is the representative point of a cluster for which the dissimilarity to all the other points of the cluster is minimal (Kaufman and Rousseeuw, 1987). By using the medoid instead of the mean, we change the algorithm from K-Means to K-Medoids (Kaufman and Rousseeuw, 1987). In K-Medoids, we can use any similarity measure, since the same distance will be used to calculate the medoids and to allocate all the points to their closest medoid to form a cluster.

Tables 5.8 and 5.11 show the results of I-Multi using different similarity measures for function benchmarks DTLZ and WFG, respectively. Similar to the analysis conducted in previous sections, the results of the global analysis are shown in Tables 5.9 and 5.10 for the IGD_p and Hypervolume of the DTLZ problems respectively and in Tables 5.12 and 5.13 for the IGD_p and Hypervolume of the WFG problems respectively.

5.3.3.1 DTLZ benchmark

When considering the DTLZ problems, in general, we can see few statistically significant differences, and when they occur it is very hard to find a pattern. The only general pattern that emerged is an increase in the statistically significant differences as the number of objectives increases. The exception of this behavior is the DTLZ2 problem, where we can clearly see that Minkowski with $M_k = 0.5$ performs the best for all numbers of objectives. DTLZ2 is a concave

Table 5.8: Mean ranks of the IGD_p as used in the Kruskal-Wallis test for the DTLZ problems. Final ranks, presented in parentheses assigned according to mean ranks.

Obj.	Metrics	DTLZ1	DTLZ2	DTLZ3	DTLZ4	DTLZ5	DTLZ6	DTLZ7
3	Euclidean	**61.57 (2.50)**	60.87 (2.50)	**69.23 (2.50)**	**56.57 (2.50)**	**61.83 (2.50)**	**61.17 (2.50)**	**67.53 (2.50)**
	Tchebycheff	**57.13 (2.50)**	65.33 (3.00)	**48.77 (2.50)**	**62.80 (2.50)**	**58.67 (2.50)**	**60.90 (2.50)**	**53.67 (2.50)**
	Minkowski (0.5)	**57.17 (2.50)**	**39.67 (1.50)**	**62.90 (2.50)**	**58.47 (2.50)**	**56.23 (2.50)**	**56.03 (2.50)**	**55.00 (2.50)**
	Minkowski (4)	**66.13 (2.50)**	76.13 (3.00)	**61.10 (2.50)**	**64.17 (2.50)**	**65.27 (2.50)**	**63.90 (2.50)**	**65.80 (2.50)**
5	Euclidean	**56.23 (2.50)**	69.90 (3.00)	**59.70 (2.50)**	**55.60 (2.50)**	**57.63 (2.50)**	**60.23 (2.50)**	**62.07 (2.50)**
	Tchebycheff	**61.50 (2.50)**	65.40 (3.00)	**62.27 (2.50)**	**51.17 (2.50)**	**63.83 (2.50)**	**52.70 (2.50)**	**59.33 (2.50)**
	Minkowski (0.5)	**69.20 (2.50)**	**35.40 (1.00)**	**66.33 (2.50)**	**68.90 (2.50)**	**52.23 (2.50)**	**64.93 (2.50)**	**65.10 (2.50)**
	Minkowski (4)	**55.07 (2.50)**	71.30 (3.00)	**53.70 (2.50)**	**66.33 (2.50)**	**68.30 (2.50)**	**64.13 (2.50)**	**55.50 (2.50)**
8	Euclidean	65.10 (2.50)	60.87 (3.00)	**66.90 (2.50)**	**62.23 (2.50)**	**58.10 (2.00)**	**61.80 (2.50)**	**64.57 (2.50)**
	Tchebycheff	**42.77 (2.00)**	80.13 (3.00)	**52.23 (2.50)**	**61.13 (2.50)**	**47.60 (2.00)**	**67.43 (2.50)**	**60.33 (2.50)**
	Minkowski (0.5)	69.87 (3.00)	**17.87 (1.00)**	**64.27 (2.50)**	**58.67 (2.50)**	83.73 (4.00)	**52.17 (2.50)**	**56.20 (2.50)**
	Minkowski (4)	64.27 (2.50)	83.13 (3.00)	**58.60 (2.50)**	**59.97 (2.50)**	**52.57 (2.00)**	**60.60 (2.50)**	**60.90 (2.50)**
10	Euclidean	**66.07 (2.50)**	61.80 (3.00)	**53.00 (2.50)**	**55.57 (2.50)**	**51.90 (2.00)**	55.63 (2.50)	**56.07 (2.50)**
	Tchebycheff	**52.80 (2.50)**	77.00 (3.00)	**64.87 (2.50)**	**70.83 (2.50)**	**49.67 (2.00)**	59.53 (2.50)	**57.47 (2.50)**
	Minkowski (0.5)	**69.17 (2.50)**	**37.60 (1.00)**	**64.37 (2.50)**	**58.20 (2.50)**	83.93 (4.00)	78.67 (3.00)	**64.40 (2.50)**
	Minkowski (4)	**53.97 (2.50)**	65.60 (3.00)	**59.77 (2.50)**	**57.40 (2.50)**	**56.50 (2.00)**	**48.17 (2.00)**	**64.07 (2.50)**
15	Euclidean	63.47 (2.50)	61.60 (3.00)	**48.43 (2.50)**	**58.50 (2.50)**	**53.03 (2.00)**	**56.13 (2.50)**	48.13 (2.00)
	Tchebycheff	59.03 (2.50)	69.10 (3.00)	**71.20 (2.50)**	**68.00 (2.50)**	**51.30 (2.00)**	**59.57 (2.50)**	97.17 (3.50)
	Minkowski (0.5)	72.70 (3.00)	**35.13 (1.00)**	**60.97 (2.50)**	**55.60 (2.50)**	90.93 (4.00)	**73.10 (2.50)**	**20.10 (1.00)**
	Minkowski (4)	**46.80 (2.00)**	76.17 (3.00)	**61.40 (2.50)**	**59.90 (2.50)**	**46.73 (2.00)**	**53.20 (2.50)**	76.60 (3.50)
20	Euclidean	**67.33 (2.50)**	57.40 (2.50)	**60.60 (2.50)**	**50.80 (2.00)**	**44.57 (2.00)**	**63.63 (2.50)**	47.07 (2.00)
	Tchebycheff	**52.57 (2.50)**	77.17 (3.00)	**58.53 (2.50)**	62.03 (2.50)	**45.80 (2.00)**	**63.73 (2.50)**	69.53 (3.00)
	Minkowski (0.5)	**69.70 (2.50)**	**34.87 (1.50)**	**64.67 (2.50)**	**50.00 (2.00)**	99.07 (4.00)	**59.03 (2.50)**	**34.87 (1.50)**
	Minkowski (4)	**52.40 (2.50)**	72.57 (3.00)	**58.20 (2.50)**	79.17 (3.50)	**52.57 (2.00)**	**55.60 (2.50)**	90.53 (3.50)

Table 5.9: Overall ranks of the IGD_p as used in the Friedman test for all the DTLZ problems and numbers of objectives. Final ranks, presented in parentheses assigned according to the ranks.

Euclidean	Tchebycheff	Minkowski (0.5)	Minkowski (4)
99.0 (2.5)	**106.0 (2.5)**	**103.0 (2.5)**	**112.0 (2.5)**

Table 5.10: Overall ranks of the Hypervolume as used in the Friedman test for all the DTLZ problems and numbers of objectives. Final ranks, presented in parentheses assigned according to the ranks.

Euclidean	Tchebycheff	Minkowski (0.5)	Minkowski (4)
95.0 (2.5)	**100.5 (2.5)**	**123.0 (2.5)**	**101.5 (2.5)**

shaped function that does not poses further challenge to optimization algorithms (other than its unusual shape) (Deb et al., 2002), hence the optimizers usually find many non-dominated solutions well spread over the objective space. This high number of well spread solutions can increase the influence of the clustering mechanism in the final result of the optimizer, since it is easier to distinguish between different clustering mechanisms in such scenario.

In an overall view of the results from the summarized Tables 5.9 and 5.10 considering the IGD_p and Hypervolume results respectively, there is no significant difference between any of the algorithms. Although the overall ranking attributed to the Euclidean distance is slightly smaller than the others.

5.3.3.2 WFG benchmark

Considering the results for the WFG benchmark functions, we can identify more significant differences than for the DTLZ. As in the previous case, in general, the differences increase with the number of objectives, but besides that behavior it is very hard to find patterns. Exceptions are the functions WFG3 and WFG4, where for $m > 8$, Minkowski with $M_k = 0.5$ achieves the best results in all cases.

Table 5.11: Mean ranks of the IGD_p as used in the Kruskal-Wallis test for the WFG problems. Final ranks, presented in parentheses assigned according to mean ranks.

Obj.	Metrics	WFG1	WFG2	WFG3	WFG4	WFG5	WFG6	WFG7	WFG8	WFG9
3	Euclidean	**62.50 (2.50)**	**61.70 (2.50)**	**59.47 (2.50)**	**48.30 (2.50)**	**69.90 (2.50)**	**53.10 (2.50)**	**60.60 (2.50)**	**48.03 (2.50)**	**64.77 (2.50)**
	Tchebycheff	**60.23 (2.50)**	**66.37 (2.50)**	**63.27 (2.50)**	**64.20 (2.50)**	**59.30 (2.50)**	**64.90 (2.50)**	**65.93 (2.50)**	**67.70 (2.50)**	**60.57 (2.50)**
	Minkowski (0.5)	**62.23 (2.50)**	**60.17 (2.50)**	**59.87 (2.50)**	**61.73 (2.50)**	**49.23 (2.50)**	**63.17 (2.50)**	**59.27 (2.50)**	**64.77 (2.50)**	**50.60 (2.50)**
	Minkowski (4)	**57.03 (2.50)**	**53.77 (2.50)**	**59.40 (2.50)**	**67.77 (2.50)**	**63.57 (2.50)**	**60.83 (2.50)**	**56.20 (2.50)**	**61.50 (2.50)**	**66.07 (2.50)**
5	Euclidean	**61.53 (2.50)**	**58.03 (2.50)**	**54.50 (2.50)**	60.63 (2.50)	**51.17 (2.00)**	**46.90 (2.50)**	**55.47 (2.00)**	61.67 (2.50)	**55.57 (2.50)**
	Tchebycheff	**52.63 (2.50)**	**66.97 (2.50)**	**74.67 (2.50)**	**43.97 (2.00)**	74.53 (3.50)	**69.27 (2.50)**	**57.93 (2.00)**	**39.37 (2.00)**	**64.33 (2.50)**
	Minkowski (0.5)	**64.70 (2.50)**	**51.50 (2.50)**	**52.73 (2.50)**	70.43 (3.00)	**51.00 (2.00)**	**65.63 (2.50)**	85.83 (4.00)	83.60 (3.50)	**65.53 (2.50)**
	Minkowski (4)	**63.13 (2.50)**	**65.50 (2.50)**	**60.10 (2.50)**	66.97 (2.50)	65.30 (2.50)	**60.20 (2.50)**	**42.77 (2.00)**	**57.37 (2.00)**	**56.57 (2.50)**
8	Euclidean	**51.83 (2.50)**	**50.47 (2.50)**	55.63 (2.50)	**66.77 (2.50)**	60.83 (2.50)	**42.67 (2.00)**	**44.60 (2.00)**	**49.80 (2.50)**	**43.63 (1.50)**
	Tchebycheff	**66.90 (2.50)**	**72.00 (2.50)**	71.30 (3.00)	**56.47 (2.50)**	**50.03 (2.00)**	61.90 (2.50)	**61.97 (2.00)**	**69.90 (2.50)**	67.10 (3.00)
	Minkowski (0.5)	**61.00 (2.50)**	**55.17 (2.50)**	**42.40 (1.50)**	**49.47 (2.50)**	74.60 (3.00)	84.30 (3.50)	87.77 (4.00)	**70.37 (2.50)**	73.50 (3.00)
	Minkowski (4)	**62.27 (2.50)**	**64.37 (2.50)**	72.67 (3.00)	**69.30 (2.50)**	56.53 (2.50)	**53.13 (2.00)**	**47.67 (2.00)**	**51.93 (2.50)**	57.77 (2.50)
10	Euclidean	**65.50 (2.50)**	**60.43 (2.50)**	70.00 (3.00)	54.63 (2.00)	58.03 (2.50)	**50.23 (2.00)**	**38.13 (1.50)**	**45.80 (2.00)**	**55.30 (2.50)**
	Tchebycheff	**58.83 (2.50)**	**52.73 (2.50)**	66.53 (3.00)	78.67 (3.50)	**54.83 (2.00)**	**55.03 (2.00)**	61.50 (2.50)	83.03 (3.50)	**62.20 (2.50)**
	Minkowski (0.5)	**58.13 (2.50)**	**65.53 (2.50)**	**38.33 (1.00)**	**39.10 (1.50)**	80.97 (3.50)	92.53 (4.00)	102.53 (4.00)	60.27 (2.50)	**59.13 (2.50)**
	Minkowski (4)	**59.53 (2.50)**	**63.30 (2.50)**	67.13 (3.00)	69.60 (3.00)	**48.17 (2.00)**	**44.20 (2.00)**	39.83 (2.00)	**52.90 (2.00)**	**65.37 (2.50)**
15	Euclidean	**51.70 (2.50)**	**48.40 (2.00)**	50.50 (2.00)	64.47 (2.50)	68.17 (3.00)	**52.67 (2.00)**	**42.87 (2.00)**	**46.00 (2.00)**	**52.27 (2.00)**
	Tchebycheff	**67.53 (2.50)**	74.83 (3.00)	86.33 (3.50)	88.03 (3.50)	**34.97 (1.50)**	**55.17 (2.00)**	**59.27 (2.00)**	65.10 (2.50)	58.00 (2.50)
	Minkowski (0.5)	**52.03 (2.50)**	54.33 (2.50)	**38.33 (1.50)**	**18.50 (1.00)**	90.00 (3.50)	84.93 (4.00)	97.23 (4.00)	69.83 (3.00)	78.67 (3.50)
	Minkowski (4)	**70.73 (2.50)**	64.43 (2.50)	66.83 (3.00)	71.00 (3.00)	48.87 (2.00)	**49.23 (2.00)**	**42.63 (2.00)**	61.07 (2.50)	**53.07 (2.00)**
20	Euclidean	**71.60 (2.50)**	**56.10 (2.00)**	44.17 (2.00)	68.03 (3.00)	74.67 (3.00)	**56.27 (2.00)**	**33.10 (1.50)**	**54.43 (2.50)**	**63.63 (2.50)**
	Tchebycheff	**58.40 (2.50)**	87.03 (4.00)	93.20 (4.00)	80.70 (3.00)	**29.00 (1.50)**	**50.63 (2.00)**	66.83 (2.50)	**64.97 (2.50)**	**51.93 (2.50)**
	Minkowski (0.5)	**51.40 (2.50)**	**39.93 (2.00)**	**39.20 (1.50)**	**16.80 (1.00)**	97.77 (4.00)	92.60 (4.00)	98.10 (4.00)	**70.23 (2.50)**	**73.80 (2.50)**
	Minkowski (4)	**60.60 (2.50)**	**58.93 (2.00)**	65.43 (2.50)	76.47 (3.00)	**40.57 (1.50)**	**42.50 (2.00)**	43.97 (2.00)	**52.37 (2.50)**	**52.63 (2.50)**

Table 5.12: Overall ranks of the IGD_p as used in the Friedman test for all the WFG problems and numbers of objectives. Final ranks, presented in parentheses assigned according to the ranks.

Euclidean	Tchebycheff	Minkowski (0.5)	Minkowski (4)
110.0 (1.5)	162.0 (3.5)	148.0 (3.0)	120.0 (2.0)

Table 5.13: Overall ranks of the Hypervolume as used in the Friedman test for all the WFG problems and numbers of objectives. Final ranks, presented in parentheses assigned according to the ranks.

Euclidean	Tchebycheff	Minkowski (0.5)	Minkowski (4)
111.0 (1.5)	148.0 (3.0)	151.0 (3.0)	130.0 (2.5)

In an overall analysis of Tables 5.12 and 5.13 considering the IGD_p and Hypervolume indicators respectively, the Euclidean distance achieved the best results. This can be explained by the fact that it is usually among the best measures in most of the cases, and even when it is not among the best, it usually is not among the worst, so in a general analysis it performs best and can be considered a stable metric.

5.3.3.3 Summary of the results

For the considered distance measures, we can state that, in general, I-Multi is not very sensitive to the choice of a similarity metric for clustering. However, the differences in performance among the similarity metrics increase with the number of objectives. Our hypotheses for these behaviors are: first the size of the Pareto front greatly increases with the number of objectives (except for functions DTLZ5, DTLZ6 and WFG3 which are degenerate), secondly because of the number of non-dominated solutions to be clustered which also greatly increases with the number of objectives.

Although most of the functions do not exhibit a clear pattern suggesting which measure is better, for DTLZ2, WFG3 and WFG4, the Minkowski metric with $M_k = 0.5$ achieved the

best results, which indicates that the selection of an appropriate similarity measure can be problem-dependent.

Finally, from the two summarized tables for both problems, we recommend using the Euclidean distance to cluster the solutions, since the results indicate that it is a robust metric which in most of the cases appears ranked among the best metrics and in very few cases was ranked among the worst metrics.

5.4 Discussion

In this chapter we investigated two important characteristics of solutions clustering strategies in optimization of multi and many-objective problems. As a representative example, we have selected I-Multi, a state-of-the-art MOPSO for MaOPs investigated in previous works (Britto et al., 2013). These characteristics were the space in which the clustering is performed and the similarity metric used to compare the solutions.

Understanding the influence of these characteristics is an important aspect for clustering-based evolutionary and swarm-based algorithms, since one can exploit this information when designing new algorithms as well as when applying them to problems with known properties, such as deception or bias.

As far as we know, this is the first study covering the impact of using different clustering spaces and similarity metrics on the performance of many-objective optimization algorithms. Since we investigated two different characteristics of the clustering phase of I-Multi, we conducted two separate experimental studies.

In the first study, we compared the results obtained by clustering the solutions in three different spaces: objectives, decision and both, which is a combination of the two aforementioned spaces. From the obtained results, we could identify that the space in which the clustering is done has an important impact on the algorithm performance. Moreover, the best choice of clustering space can be problem-dependent, and is impacted by specific properties of the problem, especially deception and bias. In an overall analysis, we can recommend clustering the solutions in the objective space, since it presented good results in most of the problems.

In the second study, we compared the results obtained when using four different clustering metrics: Euclidean, Tchebycheff, Minkowski with $M_k = 0.5$ and Minkowski with $M_k = 4$. The results indicated that, in general, the algorithm is not sensitive to the choice of the metric used. However, this sensitivity increases with the number of objectives. This choice can also be problem-dependent, but in an overall view we recommend the use of the Euclidean distance, since it was the most robust metric in our comparison.

In this study we have focused on the analysis of the K-Means algorithm and its modification to K-Medoids in the second experimental study. However, an important question is to determine if further improvement to MOPSOs could be achieved by using other clustering methods, especially those that do not require defining in advance the number of clusters. These clustering methods can generate clusters of better quality and our results indicate that the clustering quality influences the performance of the algorithm. One possible direction is to evaluate the behavior of other clustering methods that have shown good results for single-objective optimizers such as hierarchical clustering (Lozano and Larrañaga, 1999) and affinity propagation (Santana et al., 2010) algorithms.

An interesting issue is whether there exists any type of interactions between the distance metrics and the type of clustering. Is one distance metric better than the others for some clustering strategy? We have not addressed this question but it is worth considering this problem in further work.

While I-Multi is a good example of other MOPSOs, the investigation of the effect of clustering could be extended to algorithms that use probabilistic modeling of the solutions contained in each cluster. One representative example of this type of algorithms is C-Multi (Castro Jr. et al., 2016b).

Chapter 6

Hybrid competent multi-swarm based on rBOA and MOPSO

Recently a new MOPSO called I-Multi was proposed by Britto et al. (2013). I-Multi is a MOPSO designed to deal with MaOPs and uses multiple swarms to cover different areas of the objective space. Its search procedure can be divided in two phases: diversity and multi-swarm searches. In the first phase the goal is to rapidly achieve a well-diversified front, while in the second, good convergence is expected within each sub-swarm.

Since I-Multi is a recent algorithm, alternative approaches are yet to be explored in order to further improve its performance on MaOPs. In this chapter, we investigates a possible alternative for the second phase (multi-swarm search) of I-Multi focusing on competent algorithms or Estimation of Distribution Algorithms (EDAs) (Larrañaga et al., 2012). An EDA which presents good results in the literature, as well as a good documentation is the rBOA (Ahn et al., 2006), presented on Section 3.6.1.

By combining I-Multi with rBOA, we create an hybrid competent algorithm called C-Multi that joins the strengths of the I-Multi algorithm of obtaining a good diversity of solutions through the use of multi-swarms to spread the solutions across the front, with the rBOA capacity of achieving good convergence by generating solutions learned from the best solutions found so far. With this in mind the algorithm should be able to have a better performance especially in problems presenting high difficulty, with a moderate increase in the computational cost.

To calibrate important parameters of rBOA, as well as to compare it to its base algorithm and to a state-of-the-art algorithm, experimental studies were carried:

First we conducted two studies on the impact of C-Multi components. One of them evaluated the effect of the archiver used in the multi-swarm phase of the algorithm. The other assessed the effect of the number of sub-swarms.

Next, we conducted extensive experimentation to compare the performances of C-Multi, enhanced with our findings from the previous studies, and I-Multi. Moreover, we compared both algorithms to a state-of-the-art algorithm from the literature called MOEA/D-DRA (Zhang et al., 2009) in order to assess the effectiveness of the new algorithm. The results obtained indicate that C-Multi can be a competitive algorithm and the use of EDAs is a promising area in many-objective optimization.

The remainder of this chapter is organized as follows: Section 6.1 presents the C-Multi algorithm. Experimental studies to investigate the influence of components of C-Multi, as well as an empirical study comparing it to I-Multi and MOEA/D-DRA are reported in Section 6.2. Finally, Section 6.3 presents a discussion of this chapter.

6.1 Competent multi-swarm

This section presents a hybrid algorithm called Competent Multi-swarm (C-Multi). The feature that distinguishes C-Multi from the other previously introduced multi-swarm algorithms is the incorporation of a model-based search component implemented using rBOA (Ahn et al., 2006). C-Multi combines the strengths of two efficient algorithms: I-Multi (Britto et al., 2013) and rBOA. I-Multi presents a high diversity of solutions through the use of multiple swarms to spread its particles across the front. rBOA is able to achieve a good convergence by generating solutions learned from the shape of the Pareto front. By hybridizing these two algorithms, C-Multi is expected to have a better performance, especially in hard problems.

As the project of I-Multi, the project of C-Multi is based on two phases; the first uses a unique PSO to discover the different regions of the Pareto front. The second phase uses multiple swarms to specialize on a dedicate part. On each swarm, an EDA is used to focus on convergence to its allocated region.

In the first phase, a traditional SMPSO (Nebro et al., 2009) with the MGA (Laumanns and Zenklusen, 2011) archiver is used to obtain a diverse set of non-dominated solutions (basis front). Next, the multi-swarm phase begins by splitting the basis front in NS sub-fronts (set as parameter) by using the K-Means algorithm. Each sub-swarm is initialized with a seed (centroid of the cluster) and the non-dominated solutions contained in a sub-front. This seed is used to define the bounds to truncate the solutions (if necessary) to make sure each sub-swarm will explore only a limited region avoiding overlaps. Avoiding the overlap of sub-swarms is important to increase the diversity of the algorithm.

In each loop of the multi-swarm phase, the non-dominated solutions of each sub-swarm are used by rBOA to learn a model. Next, the model is sampled to create a new population (or replace a previous one) for the sub-swarm. Then, the population of the sub-swarm is added to the front if the criteria of the archiver are satisfied, ending the loop.

Algorithm 11: C-Multi

```
   // Phase 1: Diversity search
 1 F_b=Run-MGA-SMPSO()
   // Phase 2: Multi-swarm search
 2 for s=1 to SI do
 3     F=SplitFront(F_b)
 4     for k=1 to NS do
 5         for i=1 to It do
 6             P_k=rBOA(F_k)
 7             Truncate(P_k)
 8             Evaluate(P_k)
 9             F_k=Archiver(P_k)
10         end
11     end
12     F_b=Non-dominated(F)
13 end
14 return F_b
```

During the multi-swarm phase, the loops above are interrupted a predefined number of times (SI, set as parameter) to perform a split iteration. In this procedure, the fronts of all sub-swarms are merged into a single basis front (only the non-dominated solutions concerning all fronts are kept) which is split as before. Then, the multi-swarm phase is restarted. The goal of the split iterations is to allow an indirect communication between the sub-swarms in addition to eliminating duplicated solutions across the fronts. A pseudo-code of C-Multi is presented through Algorithm 11.

In this algorithm, firstly the diversity phase is conducted by running the SMPSO algorithm with the MGA archiver for a predefined number of iterations to obtain a well diversified basis front (F_b).

Next the multi-swarm stage begins. This stage is characterized by SI split iterations, where the basis front F_b is split in NS sub-fronts ($\boldsymbol{F}$=(F_1, ..., F_{NS})) by using a K-Means algorithm in the decision space as in the original I-Multi (Britto et al., 2013) algorithm. Each of these sub-fronts (F_k) is composed of a set of non-dominated solutions and a seed (the centroid of the cluster).

Then, for each front, during a predetermined number of iterations, the following procedure is executed: a model is learned for each front F_k; a new population P_k is sampled

from this model and its decision vector is truncated (if necessary) to stay within the search region around the seed; the population is evaluated through the objective function; and finally the front F_k is updated with the best solutions from the population.

After all sub-swarms were executed independently for several iterations (It), a new split iteration begins. The non-dominated solutions from all the sub-swarms are merged in the basis front F_b, which is split again into NS sub-swarms and the process repeats. At the end of the algorithm, F_b containing all the non-dominated solutions regarding all fronts, is returned as result from the algorithm.

The structure of C-Multi is basically the same as the I-Multi, however in the multi-swarm search, instead of using SMPSO, the rBOA learning method is used to create a model and sample new solutions. This replacement of the update method is used to increase the convergence capacity in this stage of the search, causing a moderate increase in the computational cost.

6.2 Empirical study

This session presents the empirical studies conducted involving the C-Multi algorithm. In the experiments, we perform an analysis of some important components of C-Multi, as the archiver used in the multi-swarm stage and the number of sub-swarms. Furthermore the performance of our algorithm is compared to its base algorithm I-Multi (Britto et al., 2013), and to the a state-of-the-art algorithm, winner of the CEC 2009 MOEA contest MOEA/D-DRA (Zhang et al., 2009).

6.2.1 Experimental setup

For the study of the impact of the components of the algorithm, i.e., the archiver (Section 6.2.2) and the number of swarms (Section 6.2.3), the problems WFG1, WFG4 and WFG6 were chosen because they are representative benchmark problems. The comparison of C-Multi to I-Multi and MOEA/D-DRA used the entire WFG family of problems to analyze the performance of the algorithms in different scenarios. The algorithms were tested using different numbers of objectives to verify how they perform as the number of objectives scales up. The main parameters used for the algorithms are summarized in Table 6.1.

Table 6.1: Parameters

C_1, C_2	varies randomly in [1.5,2.5]
Number of objectives (m)	3, 5, 8 and 10
Initial phase duration	100 iterations
Initial phase population	100 particles
Multi-swarm phase duration	100 iterations
Multi-swarm phase population	(750/number of swarms) particles
Repository maximum size	200 solutions
Multi-swarm region size	decrease from 0.5 to 0.1
Number of split iterations	5
Leader threshold (rBOA)	0.3
BIC complexity factor BIC_λ (rBOA)	0.5

The parameters C_1 and C_2, that control the effect of the personal and global best particles in the velocity, respectively, are set according to the recommendation given in the original SMPSO paper (Nebro et al., 2009), by changing these values it is possible to control the trade-off between convergence and diversity in the algorithm. The number of decision variables was set according to the recommendation given for the problems by Huband et al. (2006), decreasing or increasing the number of decision variables is an easy way of making the problem easier or

harder respectively. The complexity factor of the BIC score controls how much the algorithm penalizes the complexity of a probabilistic model, hence impacts on the trade-off between the computational cost and efficiency of the model. The complexity factor of BIC score BIC_λ, as well as the threshold used in the BEND leader algorithm (used for accomplishing mixture models in the model fitting) are set for rBOA according to the values proposed in its original paper (Ahn et al., 2006). The leader threshold of BEND algorithm used for model fitting impacts the number of clusters used in this stage, and can impact the ability of the model to efficiently modeling the dependencies between variables.

The number of split iterations, as well as the multi-swarm region size were calibrated by Britto et al. (2013) and we use the best values found, since they are representative for C-Multi as well. A split iteration can be beneficial for the search, since it removes dominated and repeated solutions as well as promoting communication among the swarms, but it can be prejudicial to the swarm, since it may reduce the number of solutions contained in each swarm, hence there are less particles to be used as guide on the search (on I-Multi) and less solutions available to be learned (on C-Multi). Regarding the multi-swarm region size, if it is increased, more overlaps are expected, hence more repeated solutions may arise among the swarms, and each swarm needs to focus its search on a larger area, weakening its specialization ability, on the other hand if we decrease this value, it can create "holes" on the front, or areas that are not covered by any swarm.

The number of iterations (initial and total), the initial size of the population, the maximum size of the repositories and the total number of particles were also set as proposed by Britto et al. (2013) since these values can be considered a good compromise between the search exploration capabilities of PSO and its computational cost. Regarding the number of particles per swarm, if the total number is not divisible by the amount of sub-swarms, the remaining particles are distributed among the sub-swarms.

To assess the quality of the Pareto fronts generated by each algorithm, two well-known quality indicators are used: The modification of the Inverted Generational Distance (IGD) known as IGD_p (Schutze et al., 2012), and the hypervolume (While et al., 2012). Up to eight objectives the exact hypervolume calculation was used, and for ten objectives an approximated version based on Monte Carlo sampling (Bader et al., 2010) was used.

The results measured using these two indicators in 30 independent runs of the algorithms for every number of objectives are compared using the Kruskal-Wallis statistical test (Kruskal and Wallis, 1952) at 5% significance level, the post-test indicates if there are statistical differences between the results and the ranks of the indicator values are used to determine the best performing algorithms.

6.2.2 Archiver of sub-swarms

In this section we compare three archivers from the literature that can be used in the multi-swarm phase of C-Multi. Crowding Distance (CD) (Deb et al., 2000), Ideal (Britto and Pozo, 2012) and Multi-level Grid Archiving (MGA) (Laumanns and Zenklusen, 2011) in order to identify which is the best in most cases.

The results of this comparison are shown in Tables 6.2 and 6.3. Regarding the IGD_p results, in problem WFG1 there were no significant differences among the archivers. In problem WFG4, significant differences were only found for eight objectives, where CD and Ideal performed better than MGA. In problem WFG6, CD had better results for three and eight objectives, while no significant differences were found for five and ten objectives.

Table 6.2: Kruskal-Wallis ranks of the IGD_p for different archivers

Obj.	Archiver	WFG1	WFG4	WFG6
3	CD	**48.43 (2.00)**	**43.53 (2.00)**	**22.90 (1.00)**
	Ideal	**45.23 (2.00)**	**44.00 (2.00)**	62.50 (2.50)
	MGA	**42.83 (2.00)**	**48.97 (2.00)**	51.10 (2.50)
5	CD	**44.00 (2.00)**	**44.20 (2.00)**	**37.80 (2.00)**
	Ideal	**45.17 (2.00)**	**44.87 (2.00)**	**49.23 (2.00)**
	MGA	**47.33 (2.00)**	**47.43 (2.00)**	**49.47 (2.00)**
8	CD	**44.53 (2.00)**	**34.47 (1.50)**	**26.07 (1.50)**
	Ideal	**45.57 (2.00)**	**29.07 (1.50)**	**36.23 (1.50)**
	MGA	**46.40 (2.00)**	72.97 (3.00)	74.20 (3.00)
10	CD	**49.77 (2.00)**	**43.83 (2.00)**	**36.37 (1.50)**
	Ideal	**44.57 (2.00)**	**39.67 (2.00)**	47.07 (2.00)
	MGA	**42.17 (2.00)**	**53.00 (2.00)**	53.07 (2.50)

Table 6.3: Kruskal-Wallis ranks of the hypervolume for different archivers

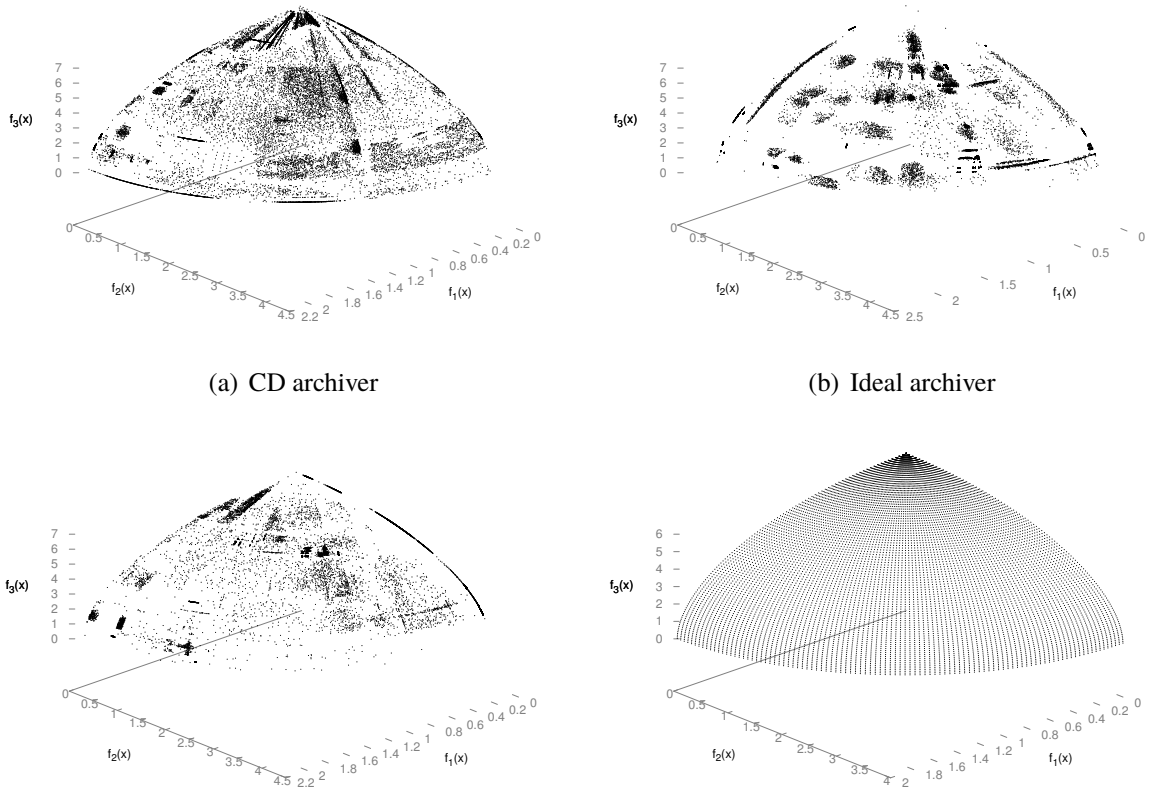
Obj.	Archiver	WFG1	WFG4	WFG6
3	CD	**49.40 (2.00)**	**35.10 (1.50)**	**24.50 (1.00)**
	Ideal	**47.83 (2.00)**	45.00 (2.00)	50.80 (2.50)
	MGA	**39.27 (2.00)**	56.40 (2.50)	61.20 (2.50)
5	CD	**45.27 (2.00)**	**25.40 (1.00)**	**44.53 (2.00)**
	Ideal	**47.37 (2.00)**	54.80 (2.50)	**43.17 (2.00)**
	MGA	**43.87 (2.00)**	56.30 (2.50)	**48.80 (2.00)**
8	CD	**49.80 (2.00)**	**16.07 (1.00)**	**23.53 (1.00)**
	Ideal	**43.03 (2.00)**	51.57 (2.00)	46.90 (2.00)
	MGA	**43.67 (2.00)**	68.87 (3.00)	66.07 (3.00)
10	CD	**43.73 (2.00)**	**31.37 (1.00)**	**37.07 (2.00)**
	Ideal	**45.30 (2.00)**	49.80 (2.50)	**47.20 (2.00)**
	MGA	**47.47 (2.00)**	55.33 (2.50)	**52.23 (2.00)**

Considering the hypervolume, no significant differences were found in WFG1. In WFG4, CD had the best results for all numbers of objectives. In WFG6, for three and eight objectives CD also had the best results, while for five and ten objectives all archivers tied.

As an overall result, considering both metrics, we can state that the archiver of sub-swarms had small influence on the search, however, in general, the archiver CD had better results in more combinations of problems and objective numbers.

These good results obtained by the CD archiver can be attributed to its characteristic of always keeping the extreme solutions of the sub-swarm. By keeping these solutions, the EDA models the region comprising these points, and has a higher probability of generating more solutions between these points. Other archivers that usually remove the extreme solutions, like Ideal, can create "holes" in the front, because the model does not learn the extremes of the front, and can be concentrated only near the center of its search region.

This behavior can be seen in Figure 6.1, where we plotted the best front (according to hypervolume) generated by C-Multi using each of the three archivers. These fronts were generated when optimizing the three objective instance of problem WFG4. Additionally, Figure 6.1(d) shows a discretization of the Pareto optimal front for this problem. Figure 6.1(a) shows a front obtained using the CD archiver, where the solutions are well-spread along the objective space. A front obtained using the Ideal archiver is presented in Figure 6.1(b) where we can see several areas of the objective space that are not covered by solutions due to the decrease on diversity. Figure 6.1(c) shows a front generated using the MGA archiver, this figure shows that this archiver is able to maintain a set of solutions well spread despite the MGA method not being able to guarantee the maintenance of the extreme solutions.



(a) CD archiver

(b) Ideal archiver

(c) MGA archiver

(d) True front

Figure 6.1: Best approximate fronts obtained using different archivers and the true front

6.2.3 Number of sub-swarms

In C-Multi, the number of swarms is an important parameter since it can impact on the content of the repository. If there are less swarms, each swarm will have more particles to cover a larger area of the front, hence we expect a higher diversity within the swarm, but since the repository size is limited, more non-dominated solutions will be discarded. On the other hand, if we increase the number of swarms, each swarm will have less particles to cover a smaller area of the front, being able to specialize in a small region, thus we expect a smaller diversity within the swarm and less solutions available for learning.

Since the model is learned from the solutions contained in the repository, its diversity and size impacts the algorithm. This section investigates this impact in the results obtained with the C-Multi algorithm and identifies the best number of sub-swarms to be used. To this end, we conducted a comparative study varying the number of sub-swarms between 10 and 100. In order to keep the same computational cost, the total number of particles on the multi-swarm phase was set to 750, then the number of particles per swarm is (750/number of swarms). If this division does not have an integer result, the remaining particles are distributed among the sub-swarms.

Since in the previous section the CD archiver presented general better results, in this study we use this archiver in the multi-swarm phase of the algorithm. The results obtained from these experiments are presented through Tables 6.4 and 6.5 where the Kruskal-Wallis ranks of the IGD_p and hypervolume are displayed.

Table 6.4: Kruskal-Wallis ranks of the IGD_p for different numbers of sub-swarms

Obj.	Swarm number	WFG1	WFG4	WFG6
3	10	225.27 (8.50)	281.87 (9.50)	271.20 (9.50)
	20	**122.63 (5.00)**	251.53 (9.00)	226.63 (9.00)
	30	**127.27 (5.00)**	204.70 (8.00)	160.27 (5.00)
	40	**116.13 (5.00)**	175.93 (6.00)	**116.27 (4.50)**
	50	**114.00 (5.00)**	121.80 (4.00)	**120.00 (4.50)**
	60	**148.80 (5.00)**	113.40 (4.00)	**104.30 (4.50)**
	70	164.23 (5.50)	111.47 (4.00)	**117.90 (4.50)**
	80	**149.07 (5.00)**	**70.50 (3.50)**	**116.73 (4.50)**
	90	173.43 (5.50)	**72.03 (3.50)**	**153.10 (4.50)**
	100	164.17 (5.50)	**101.77 (3.50)**	**118.60 (4.50)**
5	10	251.77 (9.50)	281.37 (9.00)	262.63 (9.50)
	20	187.50 (5.50)	249.77 (9.00)	228.57 (9.00)
	30	**140.43 (5.00)**	214.03 (8.00)	164.93 (5.00)
	40	**115.60 (5.00)**	164.23 (6.00)	**141.27 (4.50)**
	50	**131.17 (5.00)**	161.63 (6.00)	**141.83 (4.50)**
	60	**124.57 (5.00)**	114.33 (4.00)	**123.33 (4.50)**
	70	**145.50 (5.00)**	104.87 (4.00)	**108.93 (4.50)**
	80	**135.47 (5.00)**	**80.57 (3.00)**	**113.33 (4.50)**
	90	**139.63 (5.00)**	**73.40 (3.00)**	**115.23 (4.50)**
	100	**133.37 (5.00)**	**60.80 (3.00)**	**104.93 (4.50)**
8	10	230.53 (8.50)	277.77 (9.00)	275.53 (9.50)
	20	190.00 (7.00)	252.80 (9.00)	248.40 (9.00)
	30	155.30 (5.00)	220.93 (8.00)	181.97 (7.50)
	40	**110.00 (4.50)**	178.93 (6.50)	158.67 (5.00)
	50	**103.40 (4.50)**	162.50 (6.50)	151.50 (4.50)
	60	**110.40 (4.50)**	108.43 (4.00)	108.03 (4.00)
	70	134.77 (5.00)	**85.57 (3.00)**	102.50 (4.00)
	80	130.70 (5.00)	**85.27 (3.00)**	95.17 (4.00)
	90	169.33 (5.50)	**62.83 (3.00)**	98.43 (4.00)
	100	170.57 (5.50)	**69.97 (3.00)**	**84.80 (3.50)**
10	10	224.73 (8.50)	278.87 (9.00)	277.97 (9.50)
	20	177.80 (5.50)	245.93 (8.50)	250.17 (9.00)
	30	153.47 (5.50)	219.13 (8.00)	180.90 (6.50)
	40	**131.93 (5.00)**	180.00 (6.50)	151.17 (4.50)
	50	**150.87 (5.00)**	148.23 (5.50)	145.97 (4.50)
	60	153.40 (5.50)	112.70 (4.00)	116.00 (4.50)
	70	**148.40 (5.00)**	111.07 (4.00)	110.17 (4.50)
	80	**128.43 (5.00)**	**69.00 (3.00)**	**98.77 (4.00)**
	90	**116.80 (5.00)**	79.80 (3.50)	**82.77 (4.00)**
	100	**119.17 (5.00)**	**60.27 (3.00)**	**91.13 (4.00)**

Since we obtained different results according to problem and objective numbers, it is hard to take overall conclusions, hence we included in Tables 6.6 and 6.7 the Friedman ranks obtained for the overall analysis of the algorithms. In these tests, the average of the 30 independent runs of each subproblem (problem/objective number) is considered. A significance level of 5% is considered for this test as well.

These tables indicate that, in general, when considering the IGD_p indicator, the more swarms the better, since except from two cases (60 to 70 and 80 to 90), the Friedman ranks decrease as the number of swarms increase, also the better final ranking is obtained with 100 swarms.

Regarding the hypervolume indicator, we see a similar effect, but toward 40 swarms, where the ranks increase as the number of sub-swarms distances from 40, except between 50 and 70. Also, the best final ranking is obtained with 40 swarms.

It is known that IGD_p and hypervolume have different characteristics, as demonstrated in this section. Since each one points us to a different setting, in this work we followed the results obtained using the hypervolume due to its stronger mathematical properties.

6.2.4 C-Multi vs. I-Multi vs. MOEA/D-DRA

In this section we, compare C-Multi to I-Multi (Britto et al., 2013) and MOEA/D-DRA (Zhang et al., 2009), a state-of-the-art algorithm winner of the CEC 2009 MOEA contest.

Table 6.5: Kruskal-Wallis ranks of the hypervolume for different numbers of sub-swarms

Obj.	Swarm number	WFG1	WFG4	WFG6
3	10	209.27 (7.00)	203.17 (7.00)	258.20 (9.50)
	20	138.77 (5.50)	171.10 (5.50)	205.20 (8.50)
	30	**119.97 (5.00)**	165.90 (5.50)	155.77 (5.00)
	40	**127.83 (5.00)**	**127.20 (5.00)**	**100.27 (4.50)**
	50	**123.73 (5.00)**	**127.47 (5.00)**	**127.67 (4.50)**
	60	149.30 (5.50)	148.83 (5.50)	**115.90 (4.50)**
	70	144.27 (5.50)	**122.87 (5.00)**	**129.77 (4.50)**
	80	141.73 (5.50)	150.60 (5.50)	**128.50 (4.50)**
	90	178.37 (5.50)	132.83 (5.50)	158.43 (5.00)
	100	171.77 (5.50)	155.03 (5.50)	**125.30 (4.50)**
5	10	211.97 (7.50)	198.83 (7.50)	211.37 (7.50)
	20	156.90 (5.50)	**103.30 (3.50)**	161.17 (5.50)
	30	161.03 (5.50)	**99.83 (3.50)**	**108.57 (5.00)**
	40	**132.33 (5.00)**	**101.33 (3.50)**	148.20 (5.50)
	50	**122.30 (5.00)**	116.43 (4.00)	143.37 (5.50)
	60	**114.87 (5.00)**	156.93 (5.50)	**136.57 (5.00)**
	70	142.23 (5.50)	142.80 (5.50)	**125.83 (5.00)**
	80	162.17 (5.50)	176.63 (7.00)	**128.10 (5.00)**
	90	173.17 (5.50)	204.27 (7.50)	167.07 (5.50)
	100	**128.03 (5.00)**	204.63 (7.50)	174.77 (5.50)
8	10	**179.10 (5.50)**	144.47 (4.50)	**150.77 (5.50)**
	20	**151.43 (5.50)**	102.03 (4.00)	**181.20 (5.50)**
	30	**136.77 (5.50)**	111.20 (4.00)	**142.97 (5.50)**
	40	**153.50 (5.50)**	**78.10 (3.50)**	**152.23 (5.50)**
	50	**156.07 (5.50)**	114.90 (4.00)	**159.97 (5.50)**
	60	**135.33 (5.50)**	158.20 (6.00)	**125.03 (5.50)**
	70	**130.97 (5.50)**	149.20 (4.50)	**140.20 (5.50)**
	80	**121.67 (5.50)**	189.07 (7.50)	**159.33 (5.50)**
	90	**151.77 (5.50)**	227.83 (8.50)	**138.90 (5.50)**
	100	**188.40 (5.50)**	230.00 (8.50)	**154.40 (5.50)**
10	10	**163.42 (5.50)**	119.70 (4.00)	**173.00 (5.50)**
	20	**147.87 (5.50)**	**81.43 (3.00)**	**125.55 (5.50)**
	30	**163.90 (5.50)**	91.40 (3.50)	**140.97 (5.50)**
	40	**139.87 (5.50)**	121.07 (4.00)	**153.43 (5.50)**
	50	**145.90 (5.50)**	109.15 (4.00)	**153.08 (5.50)**
	60	**149.95 (5.50)**	158.80 (6.00)	**153.07 (5.50)**
	70	**135.77 (5.50)**	171.13 (6.50)	**142.30 (5.50)**
	80	**171.50 (5.50)**	194.97 (8.00)	**143.17 (5.50)**
	90	**123.02 (5.50)**	231.65 (8.00)	**157.30 (5.50)**
	100	**163.82 (5.50)**	225.70 (8.00)	**163.13 (5.50)**

Table 6.6: Overall Friedman ranks of the IGD_p for different numbers of sub-swarms

10	20	30	40	50	60	70	80	90	100
120.0 (9.0)	101.0 (7.5)	87.0 (6.0)	63.0 (5.0)	58.0 (5.0)	52.0 (4.5)	56.0 (5.0)	42.0 (4.5)	44.0 (4.5)	**37.0 (4.0)**

Table 6.7: Overall Friedman ranks of the Hypervolume for different numbers of sub-swarms

10	20	30	40	50	60	70	80	90	100
103.0 (7.5)	66.0 (5.5)	46.0 (5.0)	**41.0 (4.5)**	55.0 (5.5)	52.0 (5.0)	50.0 (5.0)	74.0 (5.5)	82.0 (5.5)	91.0 (6.0)

In this comparison, the entire WFG family of benchmark problems was used to provide an overview of the behavior of the algorithms as the number of objectives scales up. Since the CD archiver had the best overall results for C-Multi, we use it to prune the solutions in the multi-swarm phase of C-Multi. I-Multi used the Ideal archiver as it was designed. In order to keep a fair comparison, we used the same number of sub-swarms (40), for C-Multi and I-Multi. Also we used as stop criterion for all three algorithms the number of fitness evaluations, fixed in 85100, because I-Multi and C-Multi use a different population size than MOEA/D-DRA. The results from this experimental study are presented through Tables 6.8 and 6.9 that present the Kruskal-Wallis ranks of the IGD_p and hypervolume respectively.

Regarding the IGD_p results, for three objectives C-Multi had competitive performance, achieving the best rankings in problems WFG4, WFG6 and WFG9 and tying with I-Multi on WFG5. Besides tying with C-Multi on WFG5, I-Multi only achieved superior performance on

Table 6.8: Kruskal-Wallis ranks of the IGD_p for C-Multi, I-Multi and MOEA/D-DRA

Obj.	Algorithms	WFG1	WFG2	WFG3	WFG4	WFG5
3	C-Multi	73.33 (3.00)	39.83 (2.00)	49.17 (2.00)	**19.23 (1.00)**	**39.43 (1.50)**
	I-Multi	46.53 (2.00)	**23.03 (1.00)**	71.83 (3.00)	41.77 (2.00)	**24.33 (1.50)**
	MOEA/D-DRA	**16.63 (1.00)**	73.63 (3.00)	**15.50 (1.00)**	75.50 (3.00)	72.73 (3.00)
5	C-Multi	**40.03 (1.50)**	**31.80 (1.50)**	58.57 (2.50)	**15.57 (1.00)**	**15.73 (1.00)**
	I-Multi	**27.93 (1.50)**	**29.27 (1.50)**	62.17 (2.50)	45.43 (2.00)	45.27 (2.00)
	MOEA/D-DRA	68.53 (3.00)	75.43 (3.00)	**15.77 (1.00)**	75.50 (3.00)	75.50 (3.00)
8	C-Multi	44.83 (2.00)	**39.70 (1.50)**	48.10 (2.50)	**18.70 (1.00)**	**17.00 (1.00)**
	I-Multi	**18.20 (1.00)**	**28.97 (1.50)**	**27.00 (1.00)**	42.30 (2.00)	44.00 (2.00)
	MOEA/D-DRA	73.47 (3.00)	67.83 (3.00)	61.40 (2.50)	75.50 (3.00)	75.50 (3.00)
10	C-Multi	46.80 (2.00)	**36.43 (1.50)**	**36.03 (1.50)**	**19.77 (1.00)**	**18.70 (1.00)**
	I-Multi	**16.67 (1.00)**	42.97 (2.00)	**24.97 (1.50)**	41.23 (2.00)	42.30 (2.00)
	MOEA/D-DRA	73.03 (3.00)	57.10 (2.50)	75.50 (3.00)	75.50 (3.00)	75.50 (3.00)

Obj.	Algorithms	WFG6	WFG7	WFG8	WFG9
3	C-Multi	**16.37 (1.00)**	64.90 (2.50)	65.77 (2.50)	**19.17 (1.00)**
	I-Multi	63.13 (2.50)	56.10 (2.50)	55.23 (2.50)	49.87 (2.00)
	MOEA/D-DRA	57.00 (2.50)	**15.50 (1.00)**	**15.50 (1.00)**	67.47 (3.00)
5	C-Multi	42.43 (2.00)	**15.50 (1.00)**	**15.50 (1.00)**	**28.70 (1.50)**
	I-Multi	**18.57 (1.00)**	45.50 (2.00)	45.50 (2.00)	**32.30 (1.50)**
	MOEA/D-DRA	75.50 (3.00)	75.50 (3.00)	75.50 (3.00)	75.50 (3.00)
8	C-Multi	**26.80 (1.50)**	**15.57 (1.00)**	**15.50 (1.00)**	41.90 (2.00)
	I-Multi	**34.20 (1.50)**	45.43 (2.00)	45.50 (2.00)	**19.10 (1.00)**
	MOEA/D-DRA	75.50 (3.00)	75.50 (3.00)	75.50 (3.00)	75.50 (3.00)
10	C-Multi	**23.30 (1.50)**	**15.87 (1.00)**	**15.50 (1.00)**	44.77 (2.00)
	I-Multi	**37.70 (1.50)**	45.13 (2.00)	45.50 (2.00)	**16.23 (1.00)**
	MOEA/D-DRA	75.50 (3.00)	75.50 (3.00)	75.50 (3.00)	75.50 (3.00)

Table 6.9: Kruskal-Wallis ranks of the hypervolume for C-Multi, I-Multi and MOEA/D-DRA

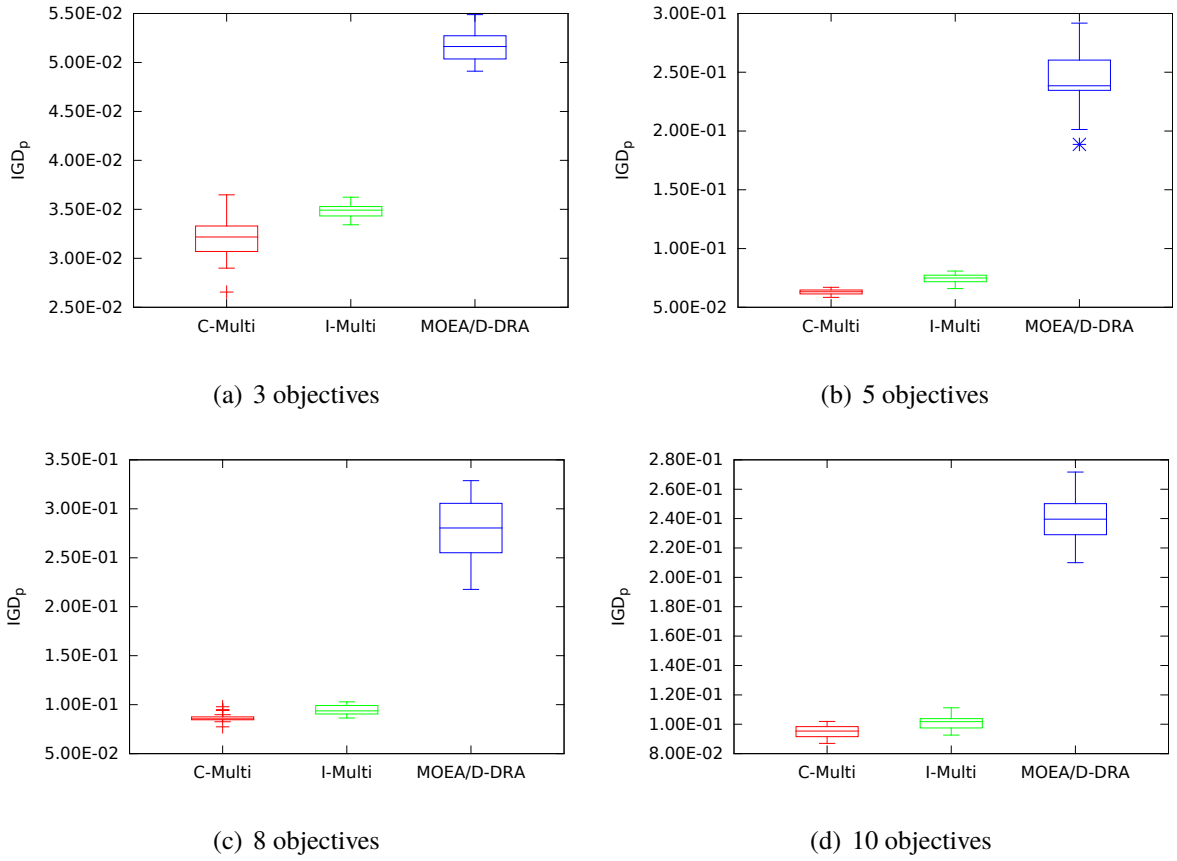
Obj.	Algorithms	WFG1	WFG2	WFG3	WFG4	WFG5
3	C-Multi	73.20 (3.00)	68.60 (3.00)	47.57 (2.00)	75.37 (3.00)	64.50 (3.00)
	I-Multi	47.03 (2.00)	52.40 (2.00)	73.43 (3.00)	**15.50 (1.00)**	**35.20 (1.50)**
	MOEA/D-DRA	**16.27 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	45.63 (2.00)	**36.80 (1.50)**
5	C-Multi	69.03 (3.00)	74.37 (3.00)	47.27 (2.00)	75.50 (3.00)	50.17 (2.00)
	I-Multi	51.93 (2.00)	46.63 (2.00)	73.73 (3.00)	44.57 (2.00)	70.83 (3.00)
	MOEA/D-DRA	**15.53 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**16.43 (1.00)**	**15.50 (1.00)**
8	C-Multi	72.87 (3.00)	71.33 (3.00)	45.50 (2.00)	46.23 (2.00)	47.07 (2.00)
	I-Multi	47.90 (2.00)	49.67 (2.00)	75.50 (3.00)	74.77 (3.00)	73.93 (3.00)
	MOEA/D-DRA	**15.73 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**
10	C-Multi	74.10 (3.00)	69.33 (3.00)	45.50 (2.00)	45.50 (2.00)	46.53 (2.00)
	I-Multi	41.60 (2.00)	51.67 (2.00)	75.50 (3.00)	75.50 (3.00)	74.47 (3.00)
	MOEA/D-DRA	**20.80 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**

Obj.	Algorithms	WFG6	WFG7	WFG8	WFG9
3	C-Multi	**23.40 (1.00)**	75.27 (3.00)	75.27 (3.00)	**22.73 (1.00)**
	I-Multi	49.33 (2.50)	45.73 (2.00)	45.73 (2.00)	51.53 (2.50)
	MOEA/D-DRA	63.77 (2.50)	**15.50 (1.00)**	**15.50 (1.00)**	62.23 (2.50)
5	C-Multi	75.50 (3.00)	73.03 (3.00)	73.87 (3.00)	**42.90 (2.00)**
	I-Multi	**15.50 (1.00)**	47.97 (2.00)	47.13 (2.00)	**39.33 (2.00)**
	MOEA/D-DRA	45.50 (2.00)	**15.50 (1.00)**	**15.50 (1.00)**	**54.27 (2.00)**
8	C-Multi	75.50 (3.00)	46.70 (2.00)	49.27 (2.00)	74.33 (3.00)
	I-Multi	43.17 (2.00)	74.30 (3.00)	71.73 (3.00)	46.63 (2.00)
	MOEA/D-DRA	**17.83 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**15.53 (1.00)**
10	C-Multi	75.50 (3.00)	46.67 (2.00)	47.10 (2.00)	75.43 (3.00)
	I-Multi	45.50 (2.00)	74.33 (3.00)	73.90 (3.00)	40.03 (2.00)
	MOEA/D-DRA	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**21.03 (1.00)**

WFG2. MOEA/D-DRA outperformed both algorithms on problems WFG1, WFG3, WFG7 and WFG8. For five objectives the results of C-Multi were better, outperforming the other algorithms in problems WFG4, WFG5, WFG7 and WFG8 and tying with I-Multi on WFG1, WFG2 and WFG9. I-Multi, besides tying with C-Multi, outperformed the other algorithms on WFG6. MOEA/D-DRA only had better rankings on WFG3.

For eight objectives, C-Multi had better rankings than the other algorithms on WFG4, WFG5, WFG7 and WFG8, and tied with I-Multi on WFG2 and WFG6. I-Multi, besides tying with C-Multi, achieved better performance on WFG1, WFG3 and WFG9. MOEA/D-DRA did not outperform the others in any problem. For ten objectives, C-Multi only lost to I-Multi on

problems WFG1 and WFG9, and tied with it on WFG3 and WFG6. MOEA/D-DRA did not have the best rankings in any problem.



(a) 3 objectives

(b) 5 objectives

(c) 8 objectives

(d) 10 objectives

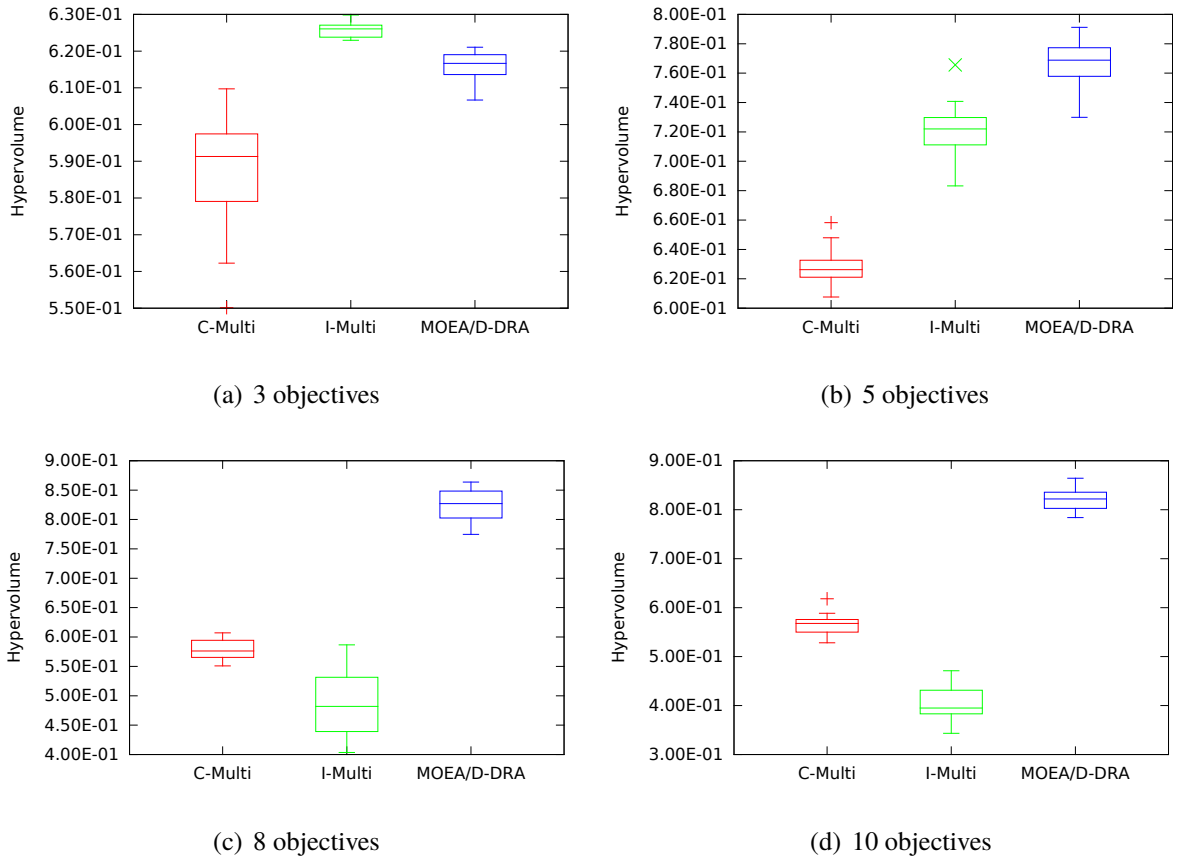
Figure 6.2: IGD_p boxplots for WFG4

Figure 6.2 shows examples of boxplots for the IGD_p results obtained by the algorithms. We only included boxplots for the WFG4 problem because it is representative of the good results of C-multi. From this figure, we can see that for all the numbers of objectives, in general, C-Multi obtained better (smaller) IGD_p results, followed closely by I-Multi. MOEA/D-DRA did not achieve good results regarding IGD_p on this problem.

Considering the hypervolume, for three objectives, C-Multi outperformed the other algorithms on WFG6 and WFG9. I-Multi had better rankings on WFG4 and tied with MOEA/D-DRA on WFG5. In the remaining problems, MOEA/D-DRA outperformed both algorithms. For five objectives, the three algorithms tied on WFG9, I-Multi achieved better results on WFG6, and MOEA/D-DRA outperformed I-Multi and C-Multi in the remaining problems. For eight and ten objectives, MOEA/D-DRA presents very good results, outperforming both algorithms in all problems.

Figure 6.3 shows examples of boxplots for the hypervolume results obtained by the algorithms. We only included boxplots for the WFG4 problem because it is representative of the good results of C-multi. From this figure we can see that, in general, MOEA/D-DRA had better (higher) hypervolume results, except for three objectives, where I-Multi performed better. For three and five objectives, I-Multi outperforms C-Multi, however for eight and ten objectives C-Multi presents better results than I-Multi.

Considering all the results presented, we can conclude that despite the excellent performance of the state-of-the-art MOEA/D-DRA regarding the hypervolume, C-Multi can



(a) 3 objectives

(b) 5 objectives

(c) 8 objectives

(d) 10 objectives

Figure 6.3: Hypervolume boxplots for WFG4

be a very competitive algorithm if we take into account a different quality indicator like IGD_p. Moreover, in general, C-Multi achieved competitive results compared to I-Multi on hypervolume, and outperformed it on IGD_p.

A possible explanation for this poor performance on hypervolume is that even using an archiver that benefits the diversity, parts of the fronts are not being properly covered by C-Multi, especially the extremes. In general, the obtained results can be considered promising, pointing to the usefulness of keep investigating the hybridization of MOPSO with EDAs for many-objective optimization.

6.3 Discussion

This chapter presented the new C-Multi algorithm designed to combine the strong points of both I-Multi and rBOA algorithms with a moderate increase in the computational cost. I-Multi is able to provide a well-spread set of solutions due to the multi-swarm strategy while achieving convergence especially in the easier problems by using an archiver that privileges convergence. rBOA is able to generate solutions close to the true Pareto front by creating a model that represents the best solutions found so far and modeling the relationships between its variables, then sampling new solutions from this model.

First we have conducted experimental studies to investigate the role of the archiving method and the number of swarms in C-Multi, and how an appropriate selection of these components can impact the searching capabilities of the algorithm. For the archiving method, we

have analyzed how three different archivers (CD, Ideal and MGA) influence the multi-swarm phase of the algorithm. This study indicated that using the Ideal archiver generates the unwanted effect of creating "holes" in the resulting front. Therefore, it was concluded that it is better to use an archiver like CD that guarantees the maintenance of generated extreme solutions of the swarms favoring the diversity of the solutions kept in the archiver.

The second experimental study involved investigating the effect of the number of sub-swarms used in the multi-swarm phase of the search. We used ten different values for the number of sub-swarms: 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100. This study obtained conflicting results regarding the used indicators, hence we configured the algorithm according to the most widely accepted indicator, in this case hypervolume.

Finally, experiments were conducted to compare C-Multi to I-Multi and the state-of-the-art MOEA/D-DRA. To consider a variety of difficult optimization scenarios, the entire WFG family of problems for 3, 5, 8 and 10 objectives, was used. The obtained results indicate that despite not being able to outperform MOEA/D-DRA in most cases regarding hypervolume, C-Multi had very good results when considering the IGD_p indicator. Moreover, C-Multi was able to outperform I-Multi in most cases regarding IGD_p, and presented competitive results according to hypervolume. A possible explanation for this poor performance on hypervolume is that part of the fronts are not being properly covered by C-Multi, especially the extremes, and hypervolume is usually sensitive to the lack of extreme solutions.

These results encourage further research on the hybridization between EDAs and MOPSOs to take advantage of their different capabilities to design more adaptive algorithms, able to explore the diverse scenarios that can be found in real-world many-objective optimization. Future works include using other EDAs to further improve convergence and the combination of decomposition and reference points to promote diversity, especially in the extremes of the search space.

Chapter 7

CMA-ES approaches for multi objective problems

In this chapter we investigate different approaches for creating multi-objective algorithms that take advantage of the CMA-ES probabilistic modeling. We have used as a basis of our work the single-objective variant of CMA-ES (Hansen and Ostermeier, 1996) as well as two of its multi-objective variants (Krause et al., 2016; Voß et al., 2010).

Our first approach is the most straightforward, where we proposed a Pareto based MO-CMA-ES variant based on a single CMA-ES model. In order to apply this model to multi-objective problems, several strategies were proposed to rank and weight the solutions to be used in the standard CMA-ES equations. This approach was able to outperform three classical MOEAs from the literature in hard bi-objective optimization problems, however it was outperformed, in general, when using the well-known DTLZ problems. This single model Pareto based approach is presented in Section 7.1.

In order to improve the results obtained in the first section, we propose a second approach, where we employ one CMA-ES model per individual of the population. Following this approach, we created two algorithm variants: the first of them was named MO-CMA-ES-rankMu and uses a rank mu update mechanism, where each individual uses their closest $\mu = T$ neighbors to update its model. The second one was named MO-CMA-ES-rankOne and uses a rank one update, where the model of each individual is updated based in the success of its offspring. The proposed variants achieved competitive results, and even outperformed the traditional IBEA (Zitzler and Künzli, 2004) algorithm in most of our test cases. This multiple model Pareto based approach is presented in Section 7.2.

Despite the good results obtained using MO-CMA-ES-rankMu and MO-CMA-ES-rankOne, Pareto based approaches are known for facing problems when dealing with problems having more than three objectives. This difficulty lead us to the development of a multi-objective CMA-ES based on decomposition. However instead of merging the CMA-ES with MOEA/D as previously done in the literature (Zapotecas-Martínez et al., 2015), we merged it with the MOEA/DD (Li et al., 2015), a state-of-the-art algorithm based on dominance and decomposition simultaneously, created especially for optimizing many-objective problems. This variant was named MOEA/DD-CMA and, as expected, was outperformed by MO-CMA-ES-rankMu and MO-CMA-ES-rankOne in most of the two and three objective variants of the problems, however as the number of objectives increased, the performance of the Pareto based approaches deteriorated and they were outperformed by MOEA/DD-CMA in most of the investigated problems. This multiple model, dominance and decomposition based approach is presented in Section 7.3.

To better organize the chapter, first we describe all the algorithms proposed, then we include all the experimental studies involving these algorithms in Section 7.4. Finally, Section 7.5 presents a discussion of the chapter.

7.1 Pareto based MO-CMA-ES with single model

The first steps for the model update of the standard single-objective CMA-ES is to rank the μ best solutions according to their fitness, and to weight them, so the best solutions have a larger influence in the model update. Then these ranked and weighted solutions are used to update the model using the remaining equations, where the fitness values are not taken into account anymore.

To create a multi-objective variant of CMA-ES based on its single-objective counterpart, first we need some mechanism to rank the solutions, and then, the standard equations, like the linear or logarithmic functions, can be used to weight these ranked solutions.

In this section we propose the idea of using Transfer Weight Functions (TWFs) as flexible components to manage the incorporation of information into a single model, Pareto based

MO-CMA-ES. A TWF is composed of a ranking method and a weighting function, hence new TWFs can be easily incorporated into the framework. As examples we created 10 TWFs and discuss the rationale behind their choice, their strengths and weaknesses. Moreover, an empirical study was conducted to assess the impact of each one in the performance of MO-CMA-ES. Finally, another experimental study was performed in order to validate our framework, where we compared it using the best performing TWFs with the state-of-the-art NSGA-III (Deb and Jain, 2014) algorithm.

Our work is inspired on previous results (Shakya and McCall, 2007; Valdez-Peña et al., 2009) in EDAs that show how the direct infusion of fitness information into the probabilistic models can lead to improved results on these algorithms. Some of the proposed TWFs incorporate information about the fitness of the solutions, this purposely violates some of the invariance properties[1] of CMA-ES, in particular its invariance under order preserving transformations of the objective function values. We hypothesize however, that TWFs can be an effective way to cope with many-objective problems (MaOPs). In this particular context, the invariance properties may be more difficult to fulfill and, in any case, less important than the needs for search efficiency.

To summarize the main contributions of this section, we: 1) Investigate the behavior of an informed MO-CMA-ES, whose main characteristic is the use of information about the quality of the solutions to update the covariance matrix. 2) Introduce TWFs as an effective and simple way to incorporate information from the solutions into the algorithm. Different TWFs are explained and compared in terms of their effect. 3) Conduct extensive experiments using difficult benchmarks that include MOPs of up to 20 objectives and present empirical evidence of the gains in performance that can be achieved by the introduced strategies. 4) Empirically compare the best performing TWFs with classical MOEAs from the literature and to CMA-ES variants of these MOEAs to validate our framework.

The next section describes in detail the characteristics of our approach to define transfer weight functions.

7.1.1 Learning strategies for MO-CMA-ES

CMA-ES was proposed to learn from good solutions of a given problem and model the path that leads to them, so it can sample new solutions closer to the optimum. It was created with single-objective problems in mind, and in these problems it is trivial to determine good and bad solutions, one just has to look at the fitness values.

Our MO-CMA-ES variant is inspired in the original single objective CMA-ES and in the first MO-CMA-ES proposed by Igel et al. (2007a). As in the single objective version, we keep only one probabilistic model for the entire search and update it using good solutions from the population and/or an external ranked and weighted archive. However, since we are dealing with multi and many-objective problems, it is hard to rank between the solutions in the same way as done for single-objective problems.

There are several ways of ranking and weighting multi-objective solutions in the literature, and all of them present specific strengths and weaknesses. In this work we propose the use of TWFs as a flexible component of the MO-CMA-ES since they are composed of a

[1]CMA-ES has several invariance properties i.e., it has uniform performance on a class of functions, thus allowing the generalization of empirical results. The following invariances are found in CMA-ES: invariance against order preserving transformations of objective function values; invariance against angle preserving transformations of the search space (rotation, reflection and translation); scale invariance; invariance against invertible linear transformation of the search space.

combination of ranking and weighting methods to be used before learning the model. To model the TWFs, three questions need to be answered.

The first of these questions is which solutions should be used in the CMA-ES model update? A discussion of this question, including the advantages and drawbacks of some options is presented in Section 7.1.1.1. The second question is how to set the weights for the selected solutions? Some alternatives to this question are presented in Section 7.1.1.2. The third question that needs to be answered is which quality indicators to use for ranking the solutions? A few quality indicators are discussed in Section 7.1.1.3. A summary of the TWFs proposed is presented in Section 7.1.1.4

Algorithm 12: Single model MO-CMA-ES framework

```
1 P=initializePopulation(λ)
2 A=archiver(P)
3 repeat
4 |   M=modelLearning(P, A)
5 |   P=modelSampling(M)
6 |   P′=mutation(P)
7 |   A=archiver(P′,A)
8 until stopping criterion is met;
9 return: A
```

In summary, our framework works as follows: First a population of size λ is randomly initialized. Then, the best (according to some archiver) non-dominated solutions from this population are stored in the external archive.

In the main loop of the algorithm, the first step is the model learning. To learn the CMA-ES model, a set of solutions is selected from the population, repository, or both, then a TWF is used to rank and weight them. Following, the standard CMA-ES equations are used to learn the model.

Following, a new population is generated from the previously learnt model. This population is mutated, and submitted to the archiver, which stores the best non-dominated solutions from the combination of the previous archive and the new population. When a stopping criterion is met, the solutions currently stored in the repository are returned as result of the algorithm. The general framework just described is presented in Algorithm 13.

7.1.1.1 Which solutions should CMA-ES model?

The variant of MO-CMA-ES used in this section works with two sets of solutions: the population and the repository. The repository, or external archive, maintains the "best" non-dominated solutions found so far during the search.

i) A first option for selecting the solutions to update the CMA-ES model is directly using the solutions in the repository. This approach presents some advantages:

   - There are always non-dominated solutions in the repository;
   - The solutions in the repository are as good and diverse as possible;
   - At the beginning of the search (until the repository is full) all the non-dominated solutions found so far are in the repository.

However, this approach also has a few drawbacks:

- The repository can be the same for several iterations (no new solution enters), leading to over-fitting. This situation is more usual when dealing with a small objective number;
- If there are many non-dominated solutions (usual scenario in many-objective), several of them will not enter the repository, thus they will not be used for learning, wasting (perhaps valuable) information gathered during the search;
- Some bias can be introduced by the archiver in the search (i.e., preference to solutions in the extremes of the front).

The other set of solutions kept during the search is the population itself. These are the newest solutions to enter the search process. Among these solutions there are usually dominated and non-dominated solutions (regarding the repository), and at the end of the iteration some of them can be present in the repository as well.

ii) A second option to select the solutions to update the CMA-ES model, is using the non-dominated solutions (regarding the repository) from the population. This approach has some advantages:

- There are always different solutions at each iteration, so there is no risk of over-fitting;
- All the non-dominated solutions found during the search are used in the learning process once;
- It is less likely to have bias introduced by the archiver (some is still possible because the domination relation of the solutions is compared to the repository).

However, this approach has a few drawbacks as well:

- It is possible to have only dominated solutions in some iterations, especially using few objective functions;
- If the problem is biased, most of the solutions found will be concentrated in one region so there is no diversity preservation mechanism introduced, as in the repository;
- The “best” solutions found are used in the learning only once, so there is no elitism.

In order to use this second option for learning the model, we must deal with the first drawback, because since we established that the algorithm will only learn from non-dominated solutions and there are none, there is no learning and the algorithm can be stuck for several iterations. Hence, in this work, when there are no non-dominated solutions in the population, the solutions in the repository are used. This strategy also alleviates the third issue, introducing some elitism when the algorithm is not able to generate good solutions.

iii) A third option for selecting the solutions, is to merge both sets. This way we use all the non-dominated solutions at hand in a given iteration to learn the model. This approach can combine some advantages of both methods as:

- There are always non-dominated solutions to learn from;
- There are usually some different solutions at each iteration, so there is a smaller risk of over-fitting;

- All the non-dominated solutions found during the search are used in the learning process at least once.

However, this approach also exhibits some of the drawbacks of both approaches:

- Some bias can be introduced by the archiver in the search. Nevertheless, since we are using more solutions to learn, this problem is alleviated;
- The diversity preserving mechanism of the archiver will have less effect because more solutions are used for learning. In biased problems it is possible to have more solutions in some regions;
- If there are few non-dominated solutions per iteration, there is a risk of over-fitting.

iv) A fourth option for selecting the solutions is to use a mating pool, as done in traditional MOEAs, like NSGA-II (Deb et al., 2000), SPEA2 (Zitzler et al., 2002) and IBEA (Zitzler and Künzli, 2004). Each of these algorithms, ranks the solutions using a particular approach. Moreover, in NSGA-II and IBEA, only solutions from the population are selected to create the mating pool, however in SPEA2, both, the population and repository are used as sources of solutions to constitute the mating pool.

Once the set of solutions to be used as source for the mating pool is defined and the solutions are ranked, each solution to compose the mating pool is selected from this set by binary tournament. This approach can combine some advantages of the first two methods as:

- The set of solutions to learn from has a fixed size;
- Since dominated solutions are allowed, a higher diversity is expected;
- There are usually different solutions at each iteration, so there is a smaller risk of over-fitting;
- If the archiver is used, it is less likely to have bias introduced by it. Otherwise, there is no bias from the archiver.

However, this approach also exhibits some of the drawbacks of both approaches:

- Since dominated solutions are allowed, a lower convergence can be achieved;
- It is possible that the best solutions are never selected for the mating pool;
- If using only the population, there is no diversity preservation mechanism;
- It is possible to have multiple copies of the same solution.

Although an empirical study comparing the effects of using each of these four sources of information for CMA-ES is interesting, this is out of the scope of this section and can be considered for future works. Here we used the third option because it uses only non-dominated solutions, besides, it combines some advantages and alleviates some drawbacks of the first two approaches.

7.1.1.2 How to set the weights for the selected solutions?

In the single-objective CMA-ES, all the operations involving the solutions are weighted, so the best solutions contribute more to the update of the model, and this contribution decreases as the ranks of the solutions get worse. This ranking is based solely on the fitness value.

Since all the solutions were selected using a particular criterion, these can be considered equally good. So setting equal weights for all solutions is the most trivial option.

Another path to explore is to use a quality indicator in order to rank the solutions, giving more weight to the solutions that present desirable characteristics according to a given metric. Different quality indicators will be presented in the next section.

Once having ranked the solutions, the classic weighting strategies of CMA-ES can be used. Here we study two: log and linear, defined by Equations (7.1) and (7.2), respectively.

$$w_i = \frac{w'_i}{\sum_{j=1}^{\mu} w'_j}, w'_i = \ln(\mu + 0.5) - \ln i, \text{ for } i = 1, ..., \mu \tag{7.1}$$

$$w_i = \frac{w'_i}{\sum_{j=1}^{\mu} w'_j}, w'_i = \mu - i, \text{ for } i = 1, ..., \mu \tag{7.2}$$

These two approaches give higher weights to the best ranked solutions and this weight decreases at different rates as the ranks get worse.

Besides these three approaches, we also propose to use another one, where the weights are set based on the quality of a solution measured by an indicator. This approach is defined in Equation (7.3).

$$\begin{aligned}
&x_{max} = \max(x^{i:\lambda}), x_{min} = \min(x^{i:\lambda}) - \frac{\min(x^{i:\lambda})}{100} \\
&w'_i = \frac{x_i - x_{min}}{x_{max} - x_{min}} \\
&\text{if } w'_i \leq 0 : w'_i = \frac{\min(x^{i:\lambda}) - x_{min}}{x_{max} - x_{min}} \\
&\text{if } w'_i > 1 : w'_i = 1, \text{for } i = 1, ..., \mu \\
&w_i = \frac{w'_i}{\sum_{j=1}^{\mu} w'_j}
\end{aligned} \tag{7.3}$$

where $\max(x^{i:\lambda})$ is the maximum indicator value for all solutions, except for ∞, which is used in the crowding distance, and likewise $\min(x^{i:\lambda})$ is the smallest value except for zero. Instead of directly using the smallest value $\min(x^{i:\lambda})$ for x_{min}, we use a slightly smaller value. This is done to set a nonzero weight to a solution when it achieves an indicator value of zero.

In this last strategy the importance given to a solution is proportional to the value given to it by the indicator. Hence a good solution will have a bigger influence on the model than a worse one. By means of this weight-assignment strategy, we expect the indicator chosen to have a higher impact on the search.

7.1.1.3 Which quality indicators to use?

In the previous section we discussed different weighting strategies where solutions might influence the search more and others less according to their ranking or measured quality.

In multi-objective search there are many metrics for evaluating the solutions in order to rank them. We now present three popular indicators.

The first one is the Crowding Distance (CD), initially proposed by Deb et al. (2000), which is a popular diversity estimator from the MOEA literature. This metric is used to estimate the density of solutions surrounding a particular point. The method for calculating CD is detailed on Section 3.3.1.1.

The second indicator used here is another extensively used metric called hypervolume (Coello et al., 2006). The hypervolume of a set of solutions measures the size of the portion of the objective space that is dominated by those solutions. Hypervolume captures in one scalar both the closeness of the solutions to the optimal set and their spread across the objective space. The hypervolume indicator is discussed on Section 3.9.2.

The main disadvantage of the hypervolume indicator is that its calculation is computationally expensive, even the best known algorithms for computing the hypervolume have running times exponential in the number of objectives, which restricts the use of hypervolume based methods to problems with less than ten objectives (Bader et al., 2010).

In order to use the hypervolume indicator to evaluate the quality of a single solution, usually the contributing hypervolume is used, which is defined according to Equation (7.4) (Brockhoff et al., 2012).

$$C_{HV}(\boldsymbol{u}, PF, \boldsymbol{r}) = HV(PF, \boldsymbol{r}) - HV(PF \setminus \{\boldsymbol{u}\}, \boldsymbol{r}) \tag{7.4}$$

where $C_{HV}(\boldsymbol{u}, PF, \boldsymbol{r})$ reflects the influence of a single point on the quality of the approximation set.

Due to the high computational cost of calculating the hypervolume for many objectives, we used a third popular indicator called $R2$ (Brockhoff et al., 2012). It was originally proposed to assess the relative quality of two approximation sets. It is an indicator that simultaneously evaluates the convergence and diversity of a Pareto front approximation and presents a low computational cost (Brockhoff et al., 2012). The $R2$ indicator is detailed on Section 3.9.3.

Similarly to hypervolume, a contributing $R2$ can be computed through Equation (7.5) (Brockhoff et al., 2012).

$$C_{R2}(\boldsymbol{u}, PF, W, \boldsymbol{z}^*) = R2(PF, W, \boldsymbol{z}^*) - R2(PF \setminus \{\boldsymbol{u}\}, W, \boldsymbol{z}^*) \tag{7.5}$$

where $C_{R2}(\boldsymbol{u}, PF, W, \boldsymbol{z}^*)$ reflects the influence of a single point on the quality of the approximation set.

7.1.1.4 Transfer weight functions

We define a transfer weight function (TWF) as a tuple composed of a ranking strategy and a weight function that determines a mapping from the "characteristics" of the ranked solutions to the weight that is used for learning.

In the previous two subsections we have discussed the two components of TWFs. Now we summarize the alternatives presented in this section. We consider three ranking strategies:

1. CD: Based on the crowding distance

2. C_{HV}: Based on the contributing hypervolume measure
3. C_{R2}: Based on the contributing $R2$ measure.

We have used three different weighting functions:

1. Linear
2. Logarithmic
3. Metric (value defined by the normalization of the metric used for ranking).

A special case of TWF is when no preference is given to the solutions and all receive equal weights. In this case, the ranking strategy is not needed and the weighting function can be simply defined as the constant function. The feasible combination of the ranking strategies and weighting functions produces 10 different TWFs that are empirically evaluated in the section of experiments. All the TWFs investigated here are summarized in Table 7.1

Table 7.1: Summary of the TWFs investigated.

Variant	Metric	Weighting
Equal	n/a	Equal
Linear(CD)	Crowding Distance	Linear
Linear(C_{R2})	R2	Linear
Linear(C_{HV})	Hypervolume	Linear
Log(CD)	Crowding Distance	Logarithmic
Log(C_{R2})	R2	Logarithmic
Log(C_{HV})	Hypervolume	Logarithmic
Metric(CD)	Crowding Distance	Metric
Metric(C_{R2})	R2	Metric
Metric(C_{HV})	Hypervolume	Metric

We emphasize that one of the contributions of the section is introducing TWF as a means of manipulating the MO-CMA-ES behavior, particularly in situations where all solutions are non-dominated. However, in addition to the TWFs discussed in this section, we envision other possible strategies to define the weights in such a way that relevant information about the quality of the solutions is *directly* infused into the model.

7.2 Pareto based MO-CMA-ES with multiple models

The results obtained using the single model MO-CMA-ES with TWFs previously proposed are promising, especially when considering harder benchmark problems. However further ways of improving the performance of MO-CMA-ES are required to obtain competitive results in the general cases, considering the easier DTLZ problems.

One alternative to improve the results of MO-CMA-ES is to employ several CMA-ES models instead of just one. By increasing the number of models, we are able to specialize them in different areas of the search space, instead of trying to describe the entire search space with a single model.

Inspired by the framework proposed in the previous section, besides the MO-CMAES variants proposed by Voß et al. (2010) and Krause et al. (2016), in this section we propose two MO-CMA-ES variants. In both variants each individual in the population encodes its own CMA-ES model, however the mechanism used to update these models are distinct. The first variant uses a rank mu update mechanism, where each individual selects its $\mu = T$ closest neighbors to update its covariance matrix. The second variant uses a rank one update, where the model of each individual is updated based in the success of its offspring. The next section describes the two proposed algorithms.

7.2.1 Rank one and rank mu approaches

This section describes the two approaches proposed to simultaneously employ multiple CMA-ES models during the search in a multi-objective CMA-ES. Since both approaches follow the same general framework, first we describe this framework, and then the particularities of each algorithm. The framework begins by randomly initializing a population of size λ, where each individual has a CMA-ES model that is initialized to the default values. Next, only the non-dominated individuals from this initial population are kept.

Algorithm 13: RankOne and rankMu MO-CMA-ES variants

```
1  P=initializePopulation(λ)
2  P=nonDom(P)
3  repeat
4  |  Q= rankOneUpdate(P) or rankMuUpdate(P)
5  |  Q′=mutation(Q)
6  |  for k=1,...,|Q′| do
7  |  |  P=nonDom(P ∪ Q′_k)
8  |  |  P=archiver(P)
9  |  end
10 until stopping criterion is met;
11 return: P
```

In the main loop of the algorithm, first one of the proposed methods is used to generate an offspring for each individual in the population and update the covariance matrix of this individual. Following a mutation is applied to the offspring, and then each solution of the offspring population is included in the main population if it is non-dominated. An archiver can be used to keep an upper bound on the size of the population (there is no lower bound). When a stopping criterion is met, the current population is returned as result of the algorithm. The general framework just described is presented in Algorithm 13. The rank one update and the rank mu update are discussed in the following sections.

7.2.1.1 Rank mu update

The first described approach is the rank mu update because it is most similar to the TWF approach proposed in Section 7.1. Besides being based in the TWF approach, this method is also inspired on the UP-MO-CMA-ES proposed by Krause et al. (2016) where each individual encodes CMA-ES model and updates this model based on information from its neighbors. However, as currently implemented, UP-MO-CMA-ES is not able to optimize problems having more than two objective functions, since it uses the direct objective values of the first objective to rank the solutions, consequently ordering the second objective in a bi-objective problem whose solutions are mutually non-dominating.

The rank mu update procedure works as follows: it takes the solutions from the population one by one, lets say k. The first step is to create a copy of it, to be used as offspring k'. Next it calculates the Euclidean distance (in objective space) from k to all the other solutions in the population, then these other solutions are sorted in an ascending order according to their distances to k, such that the first solution of this list is k (distance=0). Following these solutions are weighted using the log distribution (other distributions are due to future work) and the first T neighbors (or the entire population, whatever is smaller) are used to update the covariance

matrix of k' using the usual single-objective CMA-ES equations. After the model update, a new solution is sampled as decision variable vector of k'.

We have decided to restrict the update of the models to a certain neighborhood T due to two reasons: first, using only the closest solutions allows a larger specialization of the models in a smaller region of the search space, thus leading to more convergence, and increasing the chances that the offspring generated fill "gaps" in the Pareto front by sampling solutions close to the center of its neighborhood. The second reason is to reduce the computational cost of using the entire population in the update for each individual at each generation, especially when we use an unbounded population.

7.2.1.2 Rank one update

The second approach proposed is closer to the first MO-CMA-ES algorithms available in the literature and is inspired in the variant presented by Voß et al. (2010), which is based in the NSGA-II (Deb et al., 2000) framework.

The rank one update procedure works as follows: it takes each solution from the population at a time as well, lets say k. First it creates a copy of this solution to be used as offspring k'. Next, a new decision vector is sampled for k' based on its model (identical to the model of its parent). The sampling equation used in this procedure is the same used in the rank mu, however instead of using the mean of the μ best solutions as base for the new solution, the old decision vector is used, as in (Voß et al., 2010).

In case k' survives[2] to the next iteration, its covariance matrix and step size are updated. If the individual k (the parent) survives as well, its step size is updated based on the information of whether its offspring k' survived or not. To update the covariance matrix and step sizes in this mechanism, a simplification of the usual equations are used as in (Voß et al., 2010), and presented as follows.

The σ value is updated as:

$$\begin{aligned} \overline{p}_{succ} &= (1 - c_p)\overline{p}_{succ} + c_p p_{succ} \\ \sigma &= \sigma \exp\left(\frac{1}{d}\frac{\overline{p}_{succ} - p_{succ}^{target}}{1 - p_{succ}^{target}}\right) \end{aligned} \tag{7.6}$$

where p_{succ} is set to 1 if the offspring is successful, and to 0 otherwise.

If $\overline{p}_{succ} < p_{thresh}$, the covariance matrix is updated as:

$$\begin{aligned} \boldsymbol{p_c} &= (1 - c_c)\boldsymbol{p_c} + \sqrt{c_c(2 - c_c)}\frac{\boldsymbol{x'_k} - \boldsymbol{x_k}}{\sigma_k} \\ \boldsymbol{C} &= (1 - c_{cov})\boldsymbol{C} + c_{cov}\boldsymbol{p_c}\boldsymbol{p_c}^T \end{aligned} \tag{7.7}$$

where $\boldsymbol{x'_k}$ is the decision vector of k', $\boldsymbol{x_k}$ is the decision vector of k and σ_k is the sigma value of the individual k.

Otherwise, the covariance matrix is updated as:

$$\begin{aligned} \boldsymbol{p_c} &= (1 - c_c)\boldsymbol{p_c} \\ \boldsymbol{C} &= (1 - c_{cov})\boldsymbol{C} + c_{cov}(\boldsymbol{p_c}\boldsymbol{p_c}^T + c_c(2 - c_c)\boldsymbol{C}) \end{aligned} \tag{7.8}$$

[2] k' is non-dominated regarding the population and is not eliminated by the archiver.

In all the equations we used the default parameters defined in (Voß et al., 2010), which are: $\lambda = 1$ for the selection, $d = 1 + \frac{n}{2\lambda}$, $p_{succ}^{target} = \frac{1}{5+\sqrt{\lambda/2}}$, $c_p = \frac{p_{succ}^{target}\lambda}{2+p_{succ}^{target}\lambda}$ for the step size control, and $c_c = \frac{2}{n+2}$, $c_{cov} = \frac{2}{n^2+6}$, $p_{thresh} = 0.44$ for the covariance matrix adaptation.

7.3 Dominance and decomposition based MO-CMA-ES

Different variants of CMA-ES have been proposed for MOPs (Igel et al., 2007a,b; Voß et al., 2008), however most of them are based on Pareto dominance as main selection criterion, more specifically they are created replacing the mutation and crossover operators of the non-dominated sorting genetic algorithm II (NSGA-II) (Deb et al., 2000) by the model learning and sampling of CMA-ES.

Recently a new multi-objective CMA-ES was proposed by Zapotecas-Martínez et al. (2015) and called MOEA/D-CMA. This algorithm was created to combine the strengths of CMA-ES with those of the multi-objective evolutionary algorithm based on decomposition (MOEA/D) (Zhang and Li, 2007), which is characterized by decomposing a MOP into several subproblems, and solving them cooperatively by defining a neighborhood relation among them.

MOEA/D-CMA maintains a fully parameterized CMA-ES instance for each subproblem and works as follows: In one iteration and for each subproblem, the algorithm samples a set of solutions that can be used to update the best solutions of its neighbors as in MOEA/D. Then, the CMA-ES parameters of each subproblem are updated with the best solutions generated by its own distribution and with solutions injected from its neighbors.

Currently, however, researchers agree that combining concepts of Pareto and decomposition can be beneficial for the search of MOEAs algorithms (Deb and Jain, 2014; Li et al., 2015). An algorithm of such type is the multi-objective Evolutionary Algorithm Based on Dominance and Decomposition (MOEA/DD) (Li et al., 2015). MOEA/DD is based on the well-known MOEA/D (Zhang and Li, 2007), however, it employs a new intricate update mechanism that brings more diversity into the search, although with a slower convergence rate.

In this section we propose to extend the MOEA/D-CMA algorithm to include the update mechanism from MOEA/DD instead of the default update mechanism of MOEA/D used in MOEA/D-CMA. By changing this update mechanism, we incorporate the idea introduced in MOEA/DD of simultaneously using dominance and decomposition during the search. Our goal here is to improve the performance of MOEA/D-CMA in the same way that MOEA/DD improved the performance of MOEA/D and to investigate possible synergies between these two approaches.

A second contribution is to conduct a thorough experimental study to evaluate the performance of both algorithms. This experimental study involves two well-known families of benchmark problems summing thirteen different problems whose objective numbers scale from two to fifteen. The study is completed with extensive statistical analysis of the results to extract sound, statistically supported conclusions about the performance of the algorithms as the number of objectives scales up.

Section 7.3.1 describes our newly proposed MOEA/DD-CMA, and different strategies to define the neighborhood among different subproblems in MOEA/DD-CMA are proposed and discussed in Section 7.3.2.

7.3.1 MOEA/DD-CMA

MOEA/DD-CMA is a combination between MOEA/DD (Li et al., 2015) and MOEA/D-CMA (Zapotecas-Martínez et al., 2015). Since MOEA/DD was already described (see Section 3.5),

next we present the MOEA/D-CMA (Zapotecas-Martínez et al., 2015). Following, we present our newly proposed MOEA/DD-CMA algorithm.

7.3.1.1 MOEA/D-CMA

The main idea of MOEA/D-CMA (Zapotecas-Martínez et al., 2015) is to take advantage of the flexibility of the MOEA/D (Zhang and Li, 2007) framework and just replace the crossover operator (traditionally simulated binary crossover (SBX) (Zhang and Li, 2007) or differential evolution (DE) (Li and Zhang, 2009)) by the CMA-ES (Hansen and Ostermeier, 1996). In MOEA/D-CMA, each subproblem has a fully parameterized CMA-ES instance, involving a mean vector, step size, covariance matrix and the corresponding evolution paths. The i^{th} individual is therefore denoted by the six-tuple $\langle \boldsymbol{p_c}^i, \boldsymbol{p_\sigma}^i, \boldsymbol{C}^i, \boldsymbol{m}^i, \sigma^i, t^i \rangle$, with t^i being an iteration counter.

The complete MOEA/D-CMA algorithm works as follows: first the algorithm is initialized as in traditional MOEA/D, moreover, a CMA-ES instance is initialized for each subproblem with the default values. In the next step the solutions are sampled, evaluated, and replace the old ones if they are better. This stage begins by checking, for each subproblem, if its CMA-ES instance fits one of the reset criteria, mentioned in Section 3.6.2. Next, λ solutions are sampled and repaired if needed. Then the solutions are evaluated, the ideal point is updated with the new solutions and these new solutions replace the best solutions from the neighborhood when needed as in traditional MOEA/D.

In the next step, the covariance matrices of each subproblem are updated using the best solutions from the neighborhood. This stage begins by selecting the best solution for the current subproblem (based on its scalarized vector) from each neighbor. Then this set of solutions is used to update the covariance matrix of the current subproblem according to the default rank mu equations of CMA-ES remembering to include the modified equation to deal with solutions injected from the neighbors.

7.3.1.2 MOEA/DD-CMA

The idea of MOEA/DD-CMA is straightforward: combining the strengths of the promising MOEA/D-CMA (Zapotecas-Martínez et al., 2015) with those of the state-of-the-art MOEA/DD (Li et al., 2015) to create a new algorithm called MOEA/DD-CMA. To accomplish this, we include the update mechanism from MOEA/DD instead of the default update mechanism of MOEA/D used in MOEA/D-CMA. By changing this update mechanism, we incorporate the idea introduced in MOEA/DD of simultaneously using dominance and decomposition during the search. The main aspects of MOEA/DD-CMA are presented in Algorithm 14.

In the initialization of MOEA/DD-CMA, first a set of well-distributed weight vectors is initialized $\boldsymbol{w} = \{\boldsymbol{w}^1, ..., \boldsymbol{w}^N\}$, where N is the number of subproblems. Following, the neighborhood structure $\boldsymbol{B}$ is defined. Next, the covariance matrices, step sizes and evolution paths of all N models are initialized to their default values. Then a set of initial solutions are randomly sampled, evaluated, and the ideal point is initialized as $\boldsymbol{z}_i^* = \min u_i, i \in \{1, ..., m\}$.

In the main loop, the first step is to check the current model for the four reset criteria previously presented in Section 3.6.2, if any criterion is satisfied, the model is reset to the default values, with the previously best solution as the new mean. After that, λ solutions are sampled per subproblem using Eq. (3.16). Then the population is updated with these solutions, one at a time by using the MOEA/DD update procedure presented in Algorithm 6. In this step the ideal vector is updated as well, along with the new solutions.

The next step is the update of the models. To accomplish this task, first we select the best solution among those generated by each neighbor of i regarding $\boldsymbol{w}_i'$. These solutions are

Algorithm 14: MOEA/DD-CMA

Input: set of weight vectors $\boldsymbol{w} = \{\boldsymbol{w}^1, ..., \boldsymbol{w}^N\}$, neighborhood size T.
Output: population P.

```
[P, B] = initialize() // Parent population P, neighborhood index set B
while stopping criterion not satisfied do
    for i = 1, ..., N do
        resetCriteria(i)
        Λ=sampleSolutions()// sample λ solutions using Eq. (3.16)
        Λ=mutation(Λ)
        foreach x^c ∈ Λ do // x^c is a sampled solution
            P=updatePopulation(P, x^c)
        end
    end
    for i = 1, ..., N do
        S=selectSolutions(B_i, P)
        updateModel(⟨p_c^i, p_σ^i, C^i, m^i, σ^i, g^i⟩, S)
    end
    t = t + 1
end
return P
```

stored in a set of solutions ($S, |S| = N$) that will be injected in the adaptation of the current model. Subsequently, we adapt the covariance matrix of i, following Equations (3.17) – (3.21), remembering to include Equation (3.22) when the solutions are injected from the other neighbors (not itself). After the stopping criterion is met, the current population P is returned as output of the algorithm.

In previous experiments, we identified that using a polynomial mutation is beneficial for the algorithm to increase diversity and avoid local optima, so a polynomial mutation is performed on 15% of the solutions.

7.3.2 Neighborhood variants for MOEA/DD-CMA

The algorithms MOEA/DD (Li et al., 2015), MOEA/D-CMA (Zapotecas-Martínez et al., 2015) and our MOEA/DD-CMA share several characteristics with their base algorithm MOEA/D (Zhang and Li, 2007). One of these characteristics is the definition of a neighborhood among the subproblems.

In all these algorithms, the neighborhood relation among the subproblems is defined at the initialization, and never changed during the algorithm run. The neighborhood structure for a subproblem i can be represented as $\boldsymbol{B}_i = \{i_1, ..., i_T\}$, such that $\{\boldsymbol{w}'_{i_1}, ..., \boldsymbol{w}'_{i_T}\}$ are the T closest weight vectors to $\boldsymbol{w}^i$ considering the Euclidean distance.

The neighborhood relation in a MOEA/D based algorithm is very important, since it defines which solutions will be used for selection, recombination and replacement. Hence such algorithms can benefit from a smarter initialization of this neighborhood, or even from a dynamic redefinition of this neighborhood as the search progresses.

In our proposed algorithm MOEA/DD-CMA, the neighborhood is not used for replacement, however it is still used for selection and recombination. Moreover, since each subproblem has a fully parameterized CMA-ES model, these models can be used to determine as neighbors, the subproblems whose models are most similar.

To explore the possibilities available by the use of different neighborhood relations, we propose four approaches for dynamic recalculating the neighborhood relation among the subproblems at each iteration of the algorithm. Moreover, we devised three control methods to ensure that the differences found are not due to other external factors. Furthermore, we compare the seven created approaches with the original neighborhood relation used in MOEA/D.

Following, we describe the eight neighborhood approaches employed:

- *N0:* The original MOEA/D approach.
- *N1:* Weighted likelihood. The quality of a model $\boldsymbol{M_j}$ regarding a subproblem i is the average log likelihood (difference between the maximum value, so smaller is better) given by $\boldsymbol{M_j}$ to all solutions in the population, weighted by the scalarized fitness of each solution using the weight vector of i ($\boldsymbol{w_i}$). This can be calculated as:

$$\begin{aligned} maxLik &= \max_{k=1}^{N}(logLikelihood(\boldsymbol{x_k}, \boldsymbol{M_j}) \\ q_{ij} &= \frac{1}{N}\sum_{k=1}^{N} scalarFitness(\boldsymbol{y_k}, \boldsymbol{w_i}) \times (maxLik - logLikelihood(\boldsymbol{x_k}, \boldsymbol{M_j})) \end{aligned} \tag{7.9}$$

  The rationale behind this approach is that we have all the solutions from the population evaluated by both: the weight vector of i and the model of j. When they both agree that a solution is good, a multiplication using small values will take place, decreasing the average value (smaller is better). When one of these indicators considers a solution bad, the multiplication will involve large values, thus contributing more to the average value (larger is worse). In this case, we use the solutions in the entire population as samples to measure how much the scalar fitness using $\boldsymbol{w_i}$ and the likelihood using M_j agree on the quality of the solutions.

- *N2:* Just the likelihood. The quality of model $\boldsymbol{M_j}$ regarding a subproblem i is just the likelihood given by model $\boldsymbol{M_j}$ to the best solution of subproblem i. This is calculated as: $q_{ij} = logLikelihood(\boldsymbol{x_i}, \boldsymbol{M_j})$
- *N3:* Correlation between likelihood and scalar value. The quality of model $\boldsymbol{M_j}$ regarding a subproblem i is the correlation between the log likelihood given by $\boldsymbol{M_j}$ to all solutions in the population, and the scalar value of each solution in the population calculated using the weight vector of i ($\boldsymbol{w_i}$). This can be calculated as:

$$\begin{aligned} \boldsymbol{likelihoods_j} &= \{logLikelihood(\boldsymbol{x_1}, \boldsymbol{M_j}), ..., logLikelihood(\boldsymbol{x_N}, \boldsymbol{M_j})\} \\ \boldsymbol{scalarFitnesses_i} &= \{scalarFitness(\boldsymbol{y_1}, \boldsymbol{w_i}), ..., scalarFitness(\boldsymbol{y_N}, \boldsymbol{w_i})\} \\ q_{ij} &= correlation(\boldsymbol{scalarFitnesses_i}, \boldsymbol{likelihoods_j}) \end{aligned} \tag{7.10}$$

  This approach is similar to $N1$, however instead of using a weighted sum of the values, the correlation is used. In this case the smaller quality value the better, because we expect a negative correlation, since the smaller the scalar values the better, and the larger the log likelihood, the better.

- *N4:* Kullback-Leibler divergence. The similarity between models $\boldsymbol{M_i}$ and $\boldsymbol{M_j}$ are measured as the Kullback-Leibler divergence between the probability functions represented by the two models. The Kullback-Leibler divergence between two multi-variate normal distributions can be calculated as follows (Kullback and Leibler, 1951):

$$D_{KL}(\mathcal{N}_i||\mathcal{N}_j) = \frac{1}{2}\left(\mathrm{tr}(\boldsymbol{C}_j^{-1}, \boldsymbol{C}_i) + (\bar{m}_j - \bar{m}_i)^T \boldsymbol{C}_j^{-1}(\bar{m}_j - \bar{m}_i) - n + \ln\left(\frac{\det \boldsymbol{C}_j}{\det \boldsymbol{C}_i}\right)\right) \tag{7.11}$$

  where n is the dimension in vector space.

- *N5:* Just a control method. The quality of subproblem j regarding subproblem i is just the Euclidean distance (in decision space) between the best solutions found so far for both subproblems.
- *N6:* Just a control method. The neighborhoods are randomly updated at each iteration.
- *N7:* Just a control method. The neighborhoods are randomly set at the initialization of the algorithm and kept this way until the end (they are not updated).

7.4 Experimental studies

This section concentrates all the experimental studies conducted in this chapter. The studies regarding the single model CMA-ES are presented in Section 7.4.1, involving a comparison between all the proposed TWFs, and a comparison of the best TWFs to classical MOEAs from the literature and their CMA-ES variants.

Section 7.4.2 includes the studies regarding the multiple model MO-CMA-ES variants, involving a comparison to the best classical MOEA (IBEA) for the DTLZ problems from the previous study. Two studies are presented, the first includes two variants of MO-CMA-ES-rankOne and MO-CMA-ES-rankMu, one having an unbounded population, and the other using the crowding distance archiver to limit the population size.

The empirical studies conducted using the MOEA/DD-CMA are presented in Section 7.4.3, where first a comparison to its base algorithm, MOEA/D-CMA is presented, then a comparison between the proposed neighborhood approaches, the control approaches and the classical (original) approach. Finally MOEA/DD-CMA is compared to the bounded population variants of MO-CMA-ES-rankOne and MO-CMA-ES-rankMu.

7.4.1 Empirical studies involving the single model MO-CMA-ES

The main goal of our experiments is to investigate how a MO-CMA-ES that uses TWFs to incorporate information about the quality of the solutions in the computation of the model, behaves in comparison to classical models of updating the covariance matrix.

We also investigate a number of related questions: 1) What is the influence of the type of quality indicator used to rank the solutions? 2) What is the influence of the weighting function? 3) How the different variants of TWFs behave when the number of objectives is increased?

The basic parameters used for all the MO-CMA-ES variants used in this section are summarized in Table 7.2.

Table 7.2: Parameters of the functions and MO-CMA-ES.

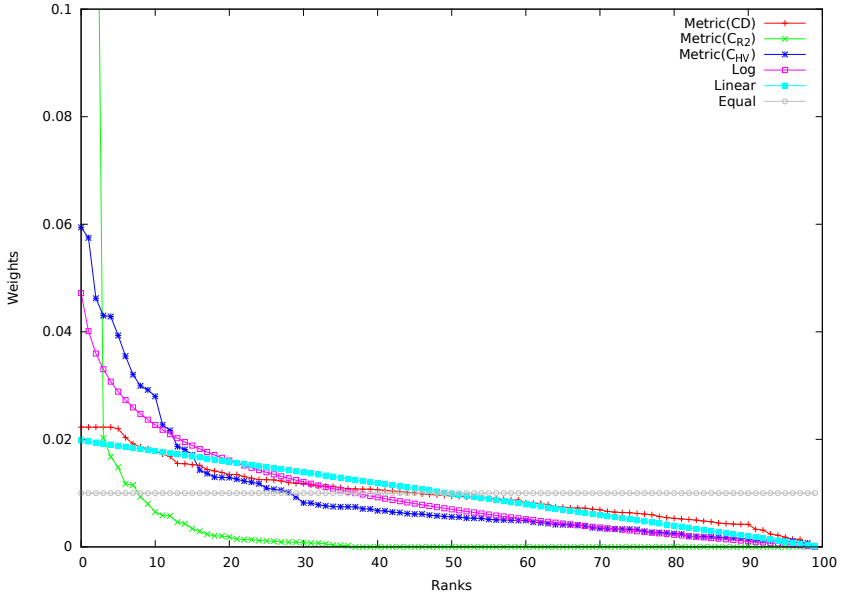
Number of objectives (m)	3, 5, 8, 10, 15 and 20
Initial population	100 solutions
Number of iterations	100
Repository maximum size	100 solutions

7.4.1.1 Example of weight distribution according to each TWF

We start by showing an example of how the weights can be set for 100 ranked solutions using different TWFs through Figure 7.1. These solutions were generated when optimizing the problem DTLZ2 with $m = 3$. In this figure we can see that each TWF distributes the weights

differently across the solutions, by doing this, the influence of a solution to the update of the model can be increased or decreased, which impacts the model learned and consequently the entire search.

Figure 7.1: Example of the computation of different weighting functions for a generation of MO-CMA-ES.



The strategies log, linear and equal always follow the same distribution since they do not consider the value of the metric to compose the weights. On the other hand C_{R2}, CD and C_{HV} can present different distributions based on the differences of the indicator values obtained by each solution. The larger the differences in weight between the solutions, the higher is the selection pressure toward the best ranked solutions. Equal does not introduce any selection pressure on the algorithm, and the selection pressure toward the best ranked solutions applied by the other weighting strategies can be expressed as: linear < log < metric.

7.4.1.2 Comparison between the algorithms up to 20 objectives

We now present the results of the experiments conducted using the two families of benchmark problems for 3, 5, 8, 10, 15 and 20 objectives. For these experiments, we used seven TWFs, six of them represent the combinations of the three weighting strategies with the two computationally cheap metrics CD and C_{R2}. The seventh TWF is equal and is added alone because it does not take into consideration the ranking methods. The TWFs using C_{HV} as metric were not compared here due to the high computational cost needed to calculate the exact hypervolume or to approximate it reliably for more than eight objectives.

The results obtained by the algorithms, measured by IGD_p are presented in Tables 7.3 and 7.4 for the DTLZ and WFG problems respectively. The results measured by Hypervolume, are presented in Tables 7.5 and 7.6, respectively. In these tables, each column represents relative performance obtained by MO-CMA-ES using one of the TWFs and each line indicates a specific number of objectives of a problem. The relative performance of a TWF is presented as the mean rank obtained by the algorithm when considering the indicator values and in parentheses the final ranking attributed according to the mean rankings and statistical differences detected by the Kruskal-Wallis test is presented. The algorithm having the smallest final ranking is highlighted.

Since it can be hard to visualize the results in the general tables and the problem families present different characteristics, summarized results are presented in Tables 7.7 and 7.8 for the DTLZ family of problems and in Tables 7.9 and 7.10 for the WFG family.

When considering the DTLZ problems, the first feature that can be appreciated in data presented is that, in general, other TWFs produce better results than equal. This behavior can be easily seen in the summarized Tables 7.7 and 7.8, where all the TWFs based on the C_{R2} indicator outperformed other TWFs more times than equal. In a more detailed analysis based in Tables 7.3, 7.4, 7.5 and 7.6, we can see that equal was among the best algorithms only in 18 out of 84 cases, from which 13 cases were for eight or fewer objectives, where the proportion of non-dominated solutions from the population is small.

Table 7.3: Mean ranks of the IGD_p as used in the Kruskal-Wallis test for the DTLZ problems up to 20 objectives. Final ranks, presented in parentheses assigned according to mean ranks.

Obj.	Algorithms	DTLZ1	DTLZ2	DTLZ3	DTLZ4	DTLZ5	DTLZ6	DTLZ7
3	Equal	122.57 (4.00)	**110.83 (3.50)**	**85.87 (4.00)**	**95.22 (4.00)**	**76.73 (3.50)**	130.73 (5.00)	138.00 (5.50)
	Linear(CD)	92.00 (4.00)	**91.37 (3.50)**	**99.87 (4.00)**	**126.40 (4.00)**	**95.33 (3.50)**	104.77 (4.50)	87.80 (3.50)
	Linear(C_{R2})	**81.60 (3.50)**	**66.47 (3.50)**	**104.57 (4.00)**	**90.23 (4.00)**	**86.20 (3.50)**	**55.60 (1.50)**	125.30 (5.00)
	Log(CD)	104.93 (4.00)	**103.73 (3.50)**	**115.63 (4.00)**	**121.13 (4.00)**	**92.30 (3.50)**	128.13 (5.00)	**63.00 (2.00)**
	Log(C_{R2})	101.60 (4.00)	**69.57 (3.50)**	**105.63 (4.00)**	**85.13 (4.00)**	**80.10 (3.50)**	**50.57 (1.50)**	121.63 (4.50)
	Metric(CD)	102.50 (4.00)	**101.03 (3.50)**	**105.53 (4.00)**	**92.02 (4.00)**	**112.33 (3.50)**	105.07 (4.50)	76.77 (2.50)
	Metric(C_{R2})	133.30 (4.50)	195.50 (7.00)	**121.40 (4.00)**	**128.37 (4.00)**	195.50 (7.00)	163.63 (6.00)	126.00 (5.00)
5	Equal	111.57 (4.50)	100.43 (4.00)	82.67 (3.00)	103.17 (4.00)	143.03 (5.00)	113.23 (4.00)	132.10 (5.00)
	Linear(CD)	124.73 (4.50)	131.07 (4.50)	122.57 (4.50)	91.27 (3.50)	110.00 (4.00)	135.97 (4.50)	74.67 (2.00)
	Linear(C_{R2})	108.43 (4.00)	96.87 (4.00)	130.13 (5.00)	**72.80 (3.00)**	102.47 (4.00)	105.57 (4.00)	123.70 (5.00)
	Log(CD)	111.20 (4.50)	134.50 (5.00)	110.47 (4.00)	136.80 (5.00)	116.47 (4.50)	103.07 (4.00)	97.27 (4.50)
	Log(C_{R2})	94.90 (4.00)	85.70 (3.50)	135.80 (5.00)	**88.30 (3.00)**	**70.00 (3.00)**	102.93 (4.00)	123.00 (5.00)
	Metric(CD)	125.43 (4.50)	122.90 (4.50)	91.07 (4.00)	102.20 (4.00)	82.23 (3.50)	108.57 (4.00)	**50.67 (1.50)**
	Metric(C_{R2})	**62.23 (2.00)**	**67.03 (2.50)**	**65.80 (2.50)**	143.97 (5.50)	114.30 (4.00)	**69.17 (3.50)**	137.10 (5.00)
8	Equal	123.00 (4.50)	124.53 (5.00)	124.97 (4.50)	**72.10 (2.50)**	144.73 (5.50)	114.83 (4.50)	**76.83 (2.00)**
	Linear(CD)	132.03 (4.50)	140.00 (5.00)	122.93 (4.50)	103.60 (3.50)	156.83 (6.00)	147.10 (5.50)	127.27 (5.50)
	Linear(C_{R2})	111.93 (4.50)	105.80 (4.50)	100.00 (4.50)	**76.60 (2.50)**	107.27 (4.00)	99.23 (4.00)	**45.50 (2.00)**
	Log(CD)	123.23 (4.50)	143.47 (5.00)	119.17 (4.50)	129.67 (5.50)	113.17 (4.50)	133.60 (5.00)	155.20 (5.50)
	Log(C_{R2})	98.93 (4.50)	66.40 (2.00)	110.70 (4.50)	81.53 (3.00)	**56.23 (2.00)**	80.33 (3.00)	**55.00 (2.00)**
	Metric(CD)	128.27 (4.50)	135.50 (5.00)	138.30 (4.50)	122.87 (5.00)	83.30 (3.00)	131.43 (5.00)	139.97 (5.50)
	Metric(C_{R2})	**21.10 (1.00)**	**22.80 (1.50)**	**22.43 (1.00)**	152.13 (6.00)	76.97 (3.00)	**31.97 (1.00)**	138.73 (5.50)
10	Equal	127.03 (4.50)	127.73 (5.00)	124.60 (4.50)	**60.43 (2.00)**	165.03 (6.00)	122.67 (4.50)	95.47 (3.50)
	Linear(CD)	133.50 (4.50)	128.23 (5.00)	132.80 (4.50)	125.03 (5.00)	133.90 (5.50)	133.93 (4.50)	150.93 (5.50)
	Linear(C_{R2})	115.23 (4.50)	105.30 (4.50)	116.50 (4.50)	86.77 (3.00)	124.10 (5.50)	111.37 (4.50)	76.37 (3.00)
	Log(CD)	116.17 (4.50)	152.90 (5.50)	115.60 (4.50)	133.07 (5.50)	108.83 (4.00)	110.43 (4.50)	117.20 (4.50)
	Log(C_{R2})	96.50 (4.50)	**59.03 (1.50)**	88.17 (4.00)	74.83 (2.50)	**58.30 (2.00)**	105.23 (4.50)	**38.47 (1.50)**
	Metric(CD)	132.20 (4.50)	141.67 (5.00)	141.80 (5.00)	116.20 (4.50)	74.27 (2.50)	130.73 (4.50)	153.20 (6.00)
	Metric(C_{R2})	**17.87 (1.00)**	**23.63 (1.50)**	**19.03 (1.00)**	142.17 (5.50)	74.07 (2.50)	**24.13 (1.00)**	106.87 (4.00)
15	Equal	109.97 (4.50)	122.13 (5.00)	116.03 (4.50)	111.80 (4.00)	156.63 (6.00)	123.30 (4.50)	191.60 (7.00)
	Linear(CD)	133.47 (4.50)	134.60 (5.00)	127.43 (4.50)	117.37 (4.00)	128.83 (5.00)	136.73 (4.50)	135.27 (5.50)
	Linear(C_{R2})	113.67 (4.50)	108.83 (4.50)	104.77 (4.50)	**77.60 (3.50)**	124.07 (4.50)	105.07 (4.50)	130.17 (5.50)
	Log(CD)	116.77 (4.50)	130.87 (5.00)	130.47 (4.50)	124.30 (4.50)	83.30 (3.50)	129.10 (4.50)	**70.03 (2.50)**
	Log(C_{R2})	94.83 (4.00)	75.20 (2.50)	121.10 (4.50)	91.67 (4.00)	**66.10 (2.50)**	98.83 (4.50)	**51.03 (2.50)**
	Metric(CD)	150.20 (5.00)	147.60 (5.00)	121.03 (4.50)	111.07 (4.00)	98.37 (3.50)	129.73 (4.50)	**78.57 (2.50)**
	Metric(C_{R2})	**19.60 (1.00)**	**19.27 (1.00)**	**17.67 (1.00)**	104.70 (4.00)	81.20 (3.00)	**15.73 (1.00)**	**81.83 (2.50)**
20	Equal	86.00 (3.50)	128.07 (5.00)	104.07 (4.50)	87.93 (4.00)	184.00 (7.00)	121.97 (4.50)	193.13 (6.50)
	Linear(CD)	124.90 (4.50)	148.43 (5.50)	118.53 (4.50)	129.63 (5.00)	123.13 (4.50)	127.00 (4.50)	**88.67 (3.00)**
	Linear(C_{R2})	119.73 (4.50)	91.53 (4.00)	119.83 (4.50)	**71.47 (2.00)**	122.60 (4.50)	104.50 (4.50)	**99.90 (3.00)**
	Log(CD)	133.83 (5.00)	131.77 (5.00)	121.40 (4.50)	132.00 (5.00)	**73.80 (2.50)**	134.67 (4.50)	**78.03 (3.00)**
	Log(C_{R2})	100.83 (4.00)	81.30 (2.50)	116.07 (4.50)	**72.73 (2.00)**	**61.00 (2.50)**	104.57 (4.50)	**67.50 (3.00)**
	Metric(CD)	155.80 (5.50)	136.60 (5.00)	141.50 (4.50)	123.77 (5.00)	78.60 (3.50)	130.23 (4.50)	**63.67 (3.00)**
	Metric(C_{R2})	**17.40 (1.00)**	**20.80 (1.00)**	**17.10 (1.00)**	120.97 (5.00)	95.37 (3.50)	**15.57 (1.00)**	147.60 (6.50)

Table 7.4: Mean ranks of the IGD_p as used in the Kruskal-Wallis test for the WFG problems up to 20 objectives. Final ranks, presented in parentheses assigned according to mean ranks.

Obj.	Algorithms	WFG1	WFG2	WFG3	WFG4	WFG5	WFG6	WFG7	WFG8	WFG9
3	Equal	**86.17 (3.00)**	**66.17 (3.00)**	78.37 (3.00)	**110.90 (3.50)**	**61.27 (2.00)**	65.07 (3.00)	**72.37 (3.00)**	73.23 (3.00)	**102.83 (4.00)**
	Linear(CD)	**93.53 (3.00)**	**70.37 (3.00)**	105.93 (4.50)	**88.43 (3.50)**	112.17 (4.50)	94.73 (3.50)	86.03 (3.50)	91.40 (3.50)	**91.37 (4.00)**
	Linear(C_{R2})	**59.20 (3.00)**	**64.73 (3.00)**	**56.07 (2.00)**	**77.43 (3.50)**	**42.90 (2.00)**	**56.77 (2.50)**	**61.47 (3.00)**	**52.53 (2.50)**	**98.90 (4.00)**
	Log(CD)	**95.17 (3.00)**	108.90 (3.50)	119.13 (4.50)	**96.63 (3.50)**	152.20 (5.50)	130.23 (4.50)	129.83 (4.50)	133.67 (5.00)	**97.80 (4.00)**
	Log(C_{R2})	**63.83 (3.00)**	**97.87 (3.00)**	**48.53 (2.00)**	**73.83 (3.50)**	85.67 (3.00)	91.70 (3.50)	93.43 (3.50)	73.57 (3.00)	**94.60 (4.00)**
	Metric(CD)	186.73 (6.50)	148.50 (6.00)	144.60 (5.50)	**98.43 (3.50)**	116.57 (4.50)	104.90 (4.00)	99.90 (3.50)	118.73 (4.00)	**122.10 (4.00)**
	Metric(C_{R2})	153.87 (6.50)	181.97 (6.50)	185.87 (6.50)	192.83 (7.00)	167.73 (6.50)	195.10 (7.00)	195.47 (7.00)	195.37 (7.00)	**130.90 (4.00)**
5	Equal	**67.73 (2.50)**	110.33 (4.00)	156.90 (5.50)	85.37 (3.50)	**71.80 (2.50)**	**92.37 (3.50)**	101.87 (3.50)	**47.37 (1.50)**	**80.10 (3.00)**
	Linear(CD)	87.07 (3.00)	134.43 (5.00)	122.60 (5.00)	90.57 (3.50)	96.00 (4.00)	**89.03 (3.50)**	96.87 (3.50)	98.93 (4.00)	100.77 (4.00)
	Linear(C_{R2})	76.80 (3.00)	**45.53 (1.50)**	74.37 (3.00)	**65.00 (3.00)**	86.67 (3.50)	**61.67 (3.50)**	**72.20 (3.00)**	50.03 (2.00)	91.17 (4.00)
	Log(CD)	117.43 (4.50)	99.60 (4.00)	111.40 (4.50)	105.50 (3.50)	135.87 (5.00)	**105.20 (3.50)**	121.47 (4.50)	152.83 (6.50)	110.13 (4.00)
	Log(C_{R2})	96.73 (3.50)	81.57 (3.00)	**27.10 (1.00)**	90.97 (3.50)	98.77 (4.00)	**94.13 (3.50)**	**66.80 (3.00)**	96.17 (3.50)	97.90 (4.00)
	Metric(CD)	152.90 (6.00)	94.37 (4.00)	137.40 (5.00)	119.90 (4.00)	129.57 (4.50)	**107.00 (3.50)**	94.10 (3.50)	100.87 (4.00)	127.43 (4.50)
	Metric(C_{R2})	139.83 (5.50)	172.67 (6.50)	108.73 (4.00)	181.20 (7.00)	119.83 (4.50)	189.10 (7.00)	185.20 (7.00)	192.30 (6.50)	131.00 (4.50)
8	Equal	**60.70 (2.50)**	128.87 (5.00)	146.20 (5.50)	**58.93 (2.00)**	**43.77 (1.00)**	**50.00 (2.00)**	135.60 (5.50)	**34.10 (1.00)**	**77.67 (3.50)**
	Linear(CD)	114.40 (4.50)	118.20 (5.00)	131.00 (5.00)	**50.80 (2.00)**	108.57 (4.50)	**58.73 (2.00)**	**80.90 (3.00)**	83.17 (3.50)	**86.23 (3.50)**
	Linear(C_{R2})	87.87 (3.50)	64.13 (2.00)	104.33 (4.00)	**45.60 (2.00)**	103.40 (4.50)	**45.00 (2.00)**	**73.63 (3.00)**	116.17 (4.50)	**95.67 (3.50)**
	Log(CD)	118.90 (4.50)	106.67 (4.50)	75.43 (3.00)	143.57 (5.50)	130.40 (4.50)	125.87 (5.00)	**81.97 (3.00)**	104.37 (4.00)	**116.00 (3.50)**
	Log(C_{R2})	105.53 (4.00)	**56.37 (1.50)**	**68.53 (2.50)**	144.23 (5.50)	110.70 (4.50)	140.87 (5.50)	**76.13 (3.00)**	130.33 (5.00)	**87.33 (3.50)**
	Metric(CD)	148.27 (5.00)	132.17 (5.00)	93.30 (3.50)	153.90 (5.50)	130.33 (4.50)	131.87 (5.00)	96.07 (3.50)	109.70 (4.00)	**108.37 (3.50)**
	Metric(C_{R2})	102.83 (4.00)	132.10 (5.00)	119.70 (4.50)	141.47 (5.50)	111.33 (4.50)	186.17 (6.50)	194.20 (7.00)	160.67 (6.00)	167.23 (7.00)
10	Equal	**78.77 (3.00)**	110.93 (4.00)	142.50 (5.50)	**41.13 (2.00)**	**58.53 (2.00)**	**48.50 (2.00)**	128.40 (4.50)	**85.17 (3.00)**	**93.63 (3.50)**
	Linear(CD)	95.70 (3.50)	116.40 (4.00)	107.43 (4.00)	**61.90 (2.00)**	102.13 (4.00)	**47.60 (2.00)**	102.60 (3.50)	**73.63 (3.00)**	**83.57 (3.50)**
	Linear(C_{R2})	101.77 (4.00)	**75.23 (3.50)**	100.50 (4.00)	**66.77 (2.00)**	110.47 (4.50)	**55.23 (2.00)**	86.40 (3.50)	**89.33 (3.00)**	**88.63 (3.50)**
	Log(CD)	129.33 (5.00)	117.60 (4.00)	**93.67 (3.00)**	149.27 (5.50)	125.17 (4.50)	124.23 (5.00)	**72.20 (3.00)**	**81.30 (3.00)**	**87.87 (3.50)**
	Log(C_{R2})	112.90 (4.00)	**74.80 (3.50)**	**82.77 (3.00)**	138.13 (5.50)	114.37 (4.50)	149.70 (5.50)	**60.93 (3.00)**	138.07 (6.00)	**101.77 (3.50)**
	Metric(CD)	144.20 (5.50)	145.87 (5.50)	**67.40 (3.00)**	161.77 (5.50)	146.13 (5.00)	129.03 (5.00)	93.23 (3.50)	100.40 (3.50)	**110.83 (3.50)**
	Metric(C_{R2})	**75.83 (3.00)**	**97.67 (3.50)**	144.23 (5.50)	119.53 (5.50)	81.70 (3.50)	184.20 (6.50)	194.73 (7.00)	170.60 (6.50)	172.20 (7.00)
15	Equal	110.07 (4.50)	**67.27 (3.00)**	122.07 (4.50)	**33.50 (1.50)**	103.13 (4.50)	**18.93 (1.00)**	123.03 (5.00)	139.37 (5.50)	**76.60 (3.50)**
	Linear(CD)	111.43 (4.50)	102.80 (4.00)	87.73 (3.50)	128.27 (5.00)	132.57 (4.50)	71.10 (3.00)	105.67 (3.50)	105.40 (4.00)	**93.90 (3.50)**
	Linear(C_{R2})	106.20 (4.50)	105.83 (4.00)	79.47 (3.50)	122.03 (5.00)	103.67 (4.50)	94.90 (3.50)	**68.67 (3.00)**	101.30 (3.50)	**89.43 (3.50)**
	Log(CD)	123.90 (4.50)	146.47 (5.50)	104.87 (3.50)	139.00 (5.00)	120.90 (4.50)	111.87 (4.00)	**74.33 (3.00)**	**58.30 (2.50)**	**119.37 (3.50)**
	Log(C_{R2})	138.53 (4.50)	96.83 (3.50)	**72.67 (3.00)**	136.60 (5.00)	109.93 (4.50)	144.67 (5.50)	**70.03 (3.00)**	89.20 (3.00)	**80.17 (3.50)**
	Metric(CD)	101.13 (4.50)	138.00 (5.00)	114.67 (4.00)	131.33 (5.00)	126.90 (4.50)	130.50 (5.00)	101.27 (3.50)	76.70 (3.00)	**110.03 (3.50)**
	Metric(C_{R2})	**47.23 (1.00)**	**81.30 (3.00)**	157.03 (6.00)	**47.77 (1.50)**	**41.40 (1.00)**	166.53 (6.00)	195.50 (7.00)	168.23 (6.50)	169.00 (7.00)
20	Equal	103.83 (4.00)	94.53 (3.50)	138.27 (4.50)	**39.37 (1.50)**	128.30 (4.50)	**17.47 (1.00)**	105.57 (3.50)	125.17 (4.00)	87.40 (3.50)
	Linear(CD)	100.37 (4.00)	136.27 (5.00)	101.03 (4.00)	101.87 (4.50)	125.53 (4.50)	72.30 (3.00)	**66.37 (3.00)**	**102.50 (3.50)**	110.53 (4.00)
	Linear(C_{R2})	109.90 (4.00)	94.53 (3.50)	94.00 (4.00)	120.60 (5.00)	116.37 (4.50)	101.20 (4.00)	**64.70 (3.00)**	**82.27 (3.50)**	88.60 (3.50)
	Log(CD)	115.30 (4.50)	145.40 (6.00)	95.93 (4.00)	154.40 (5.50)	128.67 (4.50)	102.87 (4.00)	92.90 (3.50)	**85.07 (3.50)**	117.53 (4.50)
	Log(C_{R2})	120.33 (4.50)	83.43 (3.00)	**70.13 (2.50)**	141.80 (5.00)	97.33 (4.50)	141.13 (5.00)	131.60 (5.00)	**82.13 (3.50)**	**65.63 (2.50)**
	Metric(CD)	121.77 (4.50)	151.73 (6.00)	117.30 (4.50)	145.23 (5.00)	105.40 (4.50)	130.00 (5.00)	**81.87 (3.00)**	111.43 (4.00)	115.67 (4.50)
	Metric(C_{R2})	**67.00 (2.50)**	**32.60 (1.00)**	121.83 (4.50)	**35.23 (1.50)**	**36.90 (1.00)**	173.53 (6.00)	195.50 (7.00)	149.93 (6.00)	153.13 (5.50)

Table 7.5: Mean ranks of the Hypervolume as used in the Kruskal-Wallis test for the DTLZ problems up to 20 objectives. Final ranks, presented in parentheses assigned according to mean ranks.

Obj.	Algorithms	DTLZ1	DTLZ2	DTLZ3	DTLZ4	DTLZ5	DTLZ6	DTLZ7
3	Equal	92.90 (3.50)	98.43 (4.00)	**53.77 (3.00)**	**106.12 (4.00)**	110.47 (4.00)	125.60 (4.50)	105.37 (4.00)
	Linear(CD)	**62.77 (3.00)**	85.70 (3.50)	**78.17 (3.00)**	**122.60 (4.00)**	89.57 (3.50)	97.87 (3.50)	83.30 (3.50)
	Linear(C_{R2})	**76.73 (3.00)**	**49.17 (2.00)**	**90.23 (3.00)**	**96.60 (4.00)**	73.90 (3.00)	**53.00 (2.00)**	**68.40 (3.00)**
	Log(CD)	97.20 (3.50)	119.10 (4.00)	99.80 (3.50)	**118.27 (4.00)**	93.03 (3.50)	126.43 (4.50)	112.63 (4.00)
	Log(C_{R2})	137.37 (5.00)	87.00 (3.50)	143.53 (6.00)	**81.10 (4.00)**	**52.33 (2.50)**	**56.77 (2.00)**	104.70 (4.00)
	Metric(CD)	**85.00 (3.00)**	103.60 (4.00)	**94.10 (3.00)**	**100.88 (4.00)**	123.70 (4.50)	103.10 (4.50)	116.13 (4.50)
	Metric(C_{R2})	186.53 (7.00)	195.50 (7.00)	178.90 (6.50)	**112.93 (4.00)**	195.50 (7.00)	175.73 (7.00)	147.97 (5.00)
5	Equal	85.87 (3.50)	103.50 (4.00)	**53.20 (2.00)**	97.87 (3.50)	155.63 (5.50)	111.80 (4.00)	**67.23 (2.50)**
	Linear(CD)	88.00 (3.50)	127.80 (4.50)	106.77 (4.00)	101.20 (3.50)	123.10 (5.00)	143.40 (5.00)	147.93 (6.00)
	Linear(C_{R2})	98.50 (3.50)	90.13 (4.00)	103.47 (4.00)	**69.20 (3.00)**	75.73 (2.50)	100.40 (4.00)	**47.13 (2.50)**
	Log(CD)	**64.00 (3.00)**	110.00 (4.00)	84.23 (3.00)	150.30 (6.50)	115.37 (4.50)	101.90 (4.00)	180.60 (6.00)
	Log(C_{R2})	123.83 (4.00)	**71.63 (3.00)**	138.90 (5.50)	**80.27 (3.00)**	**50.73 (2.00)**	**89.23 (3.50)**	**49.90 (2.50)**
	Metric(CD)	84.27 (3.50)	111.67 (4.00)	72.33 (3.00)	98.73 (3.50)	77.33 (3.00)	113.63 (4.00)	152.60 (6.00)
	Metric(C_{R2})	194.03 (7.00)	123.77 (4.50)	179.60 (6.50)	140.93 (5.00)	140.60 (5.50)	**78.13 (3.50)**	**93.10 (2.50)**
8	Equal	**91.07 (3.00)**	117.17 (5.00)	**103.57 (4.00)**	**73.63 (3.00)**	153.80 (6.00)	105.37 (4.50)	132.90 (5.00)
	Linear(CD)	120.30 (4.00)	151.30 (5.50)	**124.73 (4.00)**	104.50 (3.50)	160.50 (6.00)	142.90 (5.50)	131.77 (5.00)
	Linear(C_{R2})	**75.37 (3.00)**	101.20 (3.50)	**85.30 (4.00)**	**72.83 (3.00)**	94.80 (3.50)	93.90 (3.50)	113.53 (5.00)
	Log(CD)	**82.03 (3.00)**	123.07 (5.00)	**119.77 (4.00)**	128.10 (5.50)	131.10 (5.00)	134.10 (5.00)	142.33 (5.00)
	Log(C_{R2})	**78.30 (3.00)**	**63.23 (2.00)**	**84.20 (4.00)**	**80.90 (3.00)**	**31.70 (1.50)**	70.10 (2.50)	64.00 (2.00)
	Metric(CD)	138.97 (6.00)	126.40 (5.00)	**129.13 (4.00)**	113.30 (3.50)	107.30 (4.00)	135.20 (5.00)	136.33 (5.00)
	Metric(C_{R2})	152.47 (6.00)	**56.13 (2.00)**	**91.80 (4.00)**	165.23 (6.50)	59.30 (2.00)	**56.93 (2.00)**	**17.63 (1.00)**
10	Equal	**94.50 (4.00)**	124.80 (5.00)	101.70 (4.00)	**52.87 (2.00)**	167.17 (6.00)	122.07 (4.50)	121.00 (4.50)
	Linear(CD)	**101.27 (4.00)**	138.77 (5.00)	113.75 (4.00)	119.43 (4.50)	153.60 (5.50)	129.33 (4.50)	139.10 (5.00)
	Linear(C_{R2})	**100.55 (4.00)**	104.07 (5.00)	95.42 (3.50)	85.17 (3.50)	112.03 (4.50)	106.80 (4.50)	115.10 (4.50)
	Log(CD)	**98.92 (4.00)**	145.80 (5.00)	125.40 (4.50)	130.15 (5.00)	122.80 (5.00)	114.90 (4.50)	136.57 (5.00)
	Log(C_{R2})	**97.90 (4.00)**	**38.93 (1.50)**	83.40 (3.50)	78.10 (3.00)	**33.60 (1.50)**	103.17 (4.50)	76.83 (3.00)
	Metric(CD)	**114.62 (4.00)**	128.37 (5.00)	143.60 (5.50)	113.70 (4.50)	107.00 (4.00)	128.63 (4.50)	133.30 (5.00)
	Metric(C_{R2})	**130.75 (4.00)**	**57.77 (1.50)**	**75.23 (3.00)**	159.08 (5.50)	**42.30 (1.50)**	**33.60 (1.00)**	**16.60 (1.00)**
15	Equal	**87.30 (4.00)**	132.07 (4.50)	101.10 (4.50)	103.83 (4.00)	161.33 (6.00)	124.23 (4.50)	147.95 (5.50)
	Linear(CD)	**98.08 (4.00)**	141.58 (5.00)	131.13 (4.50)	114.05 (4.00)	140.50 (5.00)	127.20 (4.50)	129.15 (5.00)
	Linear(C_{R2})	**118.00 (4.00)**	106.02 (4.50)	112.65 (4.50)	**68.28 (3.00)**	109.50 (4.50)	109.20 (4.50)	94.85 (4.00)
	Log(CD)	**94.23 (4.00)**	105.53 (4.50)	118.63 (4.50)	130.17 (4.50)	112.27 (4.50)	123.00 (4.50)	126.68 (4.50)
	Log(C_{R2})	**106.77 (4.00)**	87.53 (4.00)	110.17 (4.50)	92.47 (4.00)	**52.50 (1.50)**	108.20 (4.50)	81.03 (3.00)
	Metric(CD)	**125.85 (4.00)**	129.93 (4.50)	132.07 (4.50)	98.25 (4.00)	134.73 (5.00)	131.13 (4.50)	140.00 (5.00)
	Metric(C_{R2})	**108.27 (4.00)**	**35.83 (1.00)**	**32.75 (1.00)**	131.45 (4.50)	**27.67 (1.50)**	**15.53 (1.00)**	**18.83 (1.00)**
20	Equal	**85.43 (3.50)**	141.43 (4.50)	104.03 (4.50)	81.82 (2.50)	182.23 (6.50)	125.07 (4.50)	106.92 (4.50)
	Linear(CD)	97.78 (4.00)	110.17 (4.50)	115.77 (4.50)	130.65 (5.50)	147.20 (5.00)	118.50 (4.50)	133.05 (5.00)
	Linear(C_{R2})	116.13 (4.00)	97.63 (4.00)	119.53 (4.50)	**63.07 (2.00)**	108.25 (4.50)	108.50 (4.50)	119.98 (4.50)
	Log(CD)	112.87 (4.00)	104.50 (4.00)	121.50 (4.50)	132.12 (5.50)	107.43 (4.50)	134.73 (4.50)	135.72 (5.00)
	Log(C_{R2})	119.73 (4.00)	105.47 (4.00)	117.37 (4.50)	**75.33 (2.00)**	**52.27 (1.50)**	101.87 (4.50)	74.28 (3.00)
	Metric(CD)	132.27 (5.00)	116.50 (4.50)	133.93 (4.50)	121.65 (5.00)	119.02 (4.50)	134.33 (4.50)	141.67 (5.00)
	Metric(C_{R2})	**74.28 (3.50)**	**62.80 (2.50)**	**26.37 (1.00)**	133.87 (5.50)	**22.10 (1.50)**	**15.50 (1.00)**	**26.88 (1.00)**

Table 7.6: Mean ranks of the Hypervolume as used in the Kruskal-Wallis test for the WFG problems up to 20 objectives. Final ranks, presented in parentheses assigned according to mean ranks.

Obj.	Algorithms	WFG1	WFG2	WFG3	WFG4	WFG5	WFG6	WFG7	WFG8	WFG9
3	Equal	**84.70 (3.00)**	**52.40 (2.00)**	78.43 (3.00)	**111.47 (3.50)**	**62.27 (2.00)**	60.03 (3.00)	61.20 (3.00)	**70.90 (2.50)**	**89.20 (3.50)**
	Linear(CD)	**89.07 (3.00)**	72.83 (3.00)	91.50 (3.00)	**89.37 (3.50)**	114.43 (4.50)	94.93 (3.50)	84.10 (3.00)	92.03 (3.50)	97.20 (4.00)
	Linear(C_{R2})	**55.73 (3.00)**	**51.27 (2.00)**	**52.03 (2.50)**	**73.03 (3.50)**	**41.80 (2.00)**	**55.23 (2.50)**	**56.67 (2.50)**	**50.07 (2.50)**	94.17 (4.00)
	Log(CD)	**97.67 (3.00)**	116.27 (4.50)	122.23 (4.50)	**102.30 (3.50)**	152.20 (5.50)	134.87 (4.50)	135.83 (5.00)	137.30 (5.00)	104.93 (4.00)
	Log(C_{R2})	**63.57 (3.00)**	103.97 (4.00)	**54.33 (2.50)**	**69.57 (3.50)**	77.67 (3.00)	94.37 (3.50)	98.03 (3.50)	75.40 (3.00)	93.73 (4.00)
	Metric(CD)	187.13 (6.50)	154.23 (6.00)	145.73 (5.50)	**98.20 (3.50)**	115.10 (4.50)	103.63 (4.00)	107.17 (4.00)	117.30 (4.50)	123.33 (4.00)
	Metric(C_{R2})	160.63 (6.50)	187.53 (6.50)	194.23 (7.00)	194.57 (7.00)	175.03 (6.50)	195.43 (7.00)	195.50 (7.00)	195.50 (7.00)	135.93 (4.50)
5	Equal	**46.60 (2.50)**	78.10 (2.50)	122.53 (4.50)	100.40 (3.50)	81.00 (3.00)	104.00 (3.50)	102.77 (4.00)	81.73 (3.00)	**65.30 (2.50)**
	Linear(CD)	**81.03 (2.50)**	108.50 (4.50)	97.27 (4.00)	97.57 (3.50)	107.40 (4.50)	96.90 (3.50)	100.43 (4.00)	112.67 (4.50)	117.10 (4.50)
	Linear(C_{R2})	**57.60 (2.50)**	**38.00 (2.00)**	**48.00 (1.50)**	**57.43 (2.50)**	110.00 (4.50)	**58.10 (3.00)**	**51.67 (2.00)**	**48.83 (2.00)**	99.80 (4.00)
	Log(CD)	134.97 (6.00)	127.57 (5.00)	113.73 (4.50)	110.27 (4.00)	142.67 (5.00)	101.47 (3.50)	123.03 (4.50)	153.23 (6.00)	131.37 (4.50)
	Log(C_{R2})	**80.30 (2.50)**	**60.50 (2.00)**	**25.73 (1.50)**	81.80 (3.50)	105.13 (4.50)	**67.93 (3.00)**	69.13 (3.00)	**44.30 (2.00)**	86.80 (3.50)
	Metric(CD)	181.17 (6.00)	146.80 (5.50)	151.03 (5.50)	108.93 (4.00)	134.37 (5.00)	115.20 (4.50)	96.67 (3.50)	104.73 (4.00)	136.20 (5.00)
	Metric(C_{R2})	156.83 (6.00)	179.03 (6.50)	180.20 (6.50)	182.10 (7.00)	**57.93 (1.50)**	194.90 (7.00)	194.80 (7.00)	193.00 (6.50)	101.93 (4.00)
8	Equal	**29.80 (1.50)**	74.67 (2.50)	107.70 (4.50)	64.83 (2.50)	92.73 (4.00)	**51.93 (2.00)**	65.33 (2.50)	**34.80 (1.00)**	104.00 (4.00)
	Linear(CD)	130.73 (5.00)	113.73 (4.00)	113.00 (4.50)	64.87 (2.50)	143.87 (5.00)	80.63 (2.50)	82.33 (2.50)	113.60 (4.00)	139.30 (5.50)
	Linear(C_{R2})	62.70 (2.00)	**53.97 (2.00)**	**54.30 (1.50)**	**48.70 (2.00)**	111.53 (4.50)	**37.20 (2.00)**	**55.03 (2.00)**	101.03 (4.00)	89.37 (3.00)
	Log(CD)	140.57 (5.50)	131.20 (5.50)	139.47 (5.00)	142.53 (6.00)	125.30 (4.50)	146.80 (6.00)	133.60 (5.50)	165.23 (6.50)	166.23 (6.00)
	Log(C_{R2})	118.50 (5.00)	69.90 (2.50)	**34.10 (1.50)**	96.07 (3.00)	105.80 (4.50)	98.97 (3.50)	132.13 (5.50)	84.70 (3.50)	71.77 (2.50)
	Metric(CD)	164.17 (5.50)	164.07 (6.00)	163.17 (6.00)	176.13 (6.00)	127.30 (4.50)	150.90 (6.00)	105.33 (4.00)	145.50 (5.50)	139.70 (5.50)
	Metric(C_{R2})	92.03 (3.50)	130.97 (5.50)	126.77 (5.00)	145.37 (6.00)	**31.97 (1.00)**	172.07 (6.00)	164.73 (6.00)	93.63 (3.50)	**28.13 (1.50)**
10	Equal	**32.37 (1.50)**	101.47 (4.00)	125.77 (5.00)	**39.43 (2.00)**	101.30 (4.50)	**66.68 (2.50)**	**54.13 (1.50)**	69.10 (2.50)	88.50 (3.50)
	Linear(CD)	120.82 (5.00)	127.40 (4.50)	126.37 (5.00)	69.30 (2.50)	106.37 (4.50)	**83.70 (2.50)**	100.60 (4.00)	144.03 (5.50)	133.90 (5.00)
	Linear(C_{R2})	93.13 (3.50)	**47.70 (2.00)**	59.17 (2.00)	**59.63 (2.00)**	108.87 (4.50)	**45.60 (2.50)**	79.23 (3.50)	71.00 (2.50)	108.57 (4.50)
	Log(CD)	153.38 (5.50)	147.60 (5.50)	143.23 (5.50)	159.57 (5.50)	127.20 (4.50)	161.80 (6.00)	151.47 (5.50)	158.37 (6.00)	154.47 (5.50)
	Log(C_{R2})	112.75 (4.50)	55.37 (2.50)	**38.43 (1.50)**	115.35 (4.50)	125.53 (4.50)	**76.92 (2.50)**	118.63 (4.50)	98.80 (3.50)	82.40 (3.00)
	Metric(CD)	164.60 (6.00)	176.50 (6.50)	154.43 (5.50)	178.00 (6.50)	141.43 (4.50)	163.37 (6.00)	123.17 (4.50)	161.53 (6.00)	146.10 (5.50)
	Metric(C_{R2})	61.45 (2.00)	82.47 (3.00)	91.10 (3.50)	117.22 (5.00)	**27.80 (1.00)**	140.43 (6.00)	111.27 (4.50)	**35.67 (2.00)**	**24.57 (1.00)**
15	Equal	88.20 (3.50)	79.10 (2.50)	115.97 (4.00)	**37.50 (2.00)**	116.87 (4.50)	134.57 (5.00)	126.83 (5.00)	115.13 (5.00)	81.87 (3.00)
	Linear(CD)	123.98 (4.50)	131.77 (5.50)	117.70 (4.00)	106.77 (3.50)	118.13 (4.50)	113.57 (5.00)	145.57 (5.50)	139.83 (5.00)	133.63 (5.00)
	Linear(C_{R2})	108.25 (4.50)	86.80 (3.50)	94.20 (3.50)	78.80 (3.00)	118.00 (4.50)	**54.60 (1.50)**	92.43 (3.00)	110.57 (5.00)	108.40 (4.00)
	Log(CD)	146.97 (5.00)	159.80 (6.00)	142.63 (5.50)	167.40 (6.50)	119.60 (4.50)	143.50 (5.00)	143.33 (5.50)	146.27 (5.00)	147.63 (5.50)
	Log(C_{R2})	109.57 (4.50)	69.97 (2.50)	75.53 (3.50)	119.13 (4.00)	111.53 (4.50)	**47.13 (1.50)**	56.87 (2.00)	**57.37 (1.50)**	90.03 (3.50)
	Metric(CD)	142.38 (5.00)	171.30 (6.00)	167.67 (6.50)	166.13 (6.50)	138.87 (4.50)	139.10 (5.00)	147.00 (5.50)	151.30 (5.00)	158.83 (6.00)
	Metric(C_{R2})	**19.15 (1.00)**	**39.77 (2.00)**	**24.80 (1.00)**	62.77 (2.50)	**15.50 (1.00)**	106.03 (5.00)	**26.47 (1.50)**	**18.03 (1.50)**	**18.10 (1.00)**
20	Equal	98.67 (4.50)	97.83 (3.00)	115.18 (4.50)	**49.93 (2.00)**	135.03 (4.50)	127.67 (5.00)	137.42 (5.50)	121.02 (5.00)	101.90 (4.50)
	Linear(CD)	121.47 (4.50)	145.47 (6.00)	126.27 (4.50)	110.30 (3.50)	109.75 (4.50)	114.17 (5.00)	160.35 (5.50)	143.78 (5.50)	134.50 (4.50)
	Linear(C_{R2})	114.98 (4.50)	83.77 (3.00)	100.20 (4.00)	96.73 (3.50)	118.02 (4.50)	**60.57 (1.50)**	86.27 (2.50)	81.20 (3.00)	97.13 (3.50)
	Log(CD)	144.77 (4.50)	156.30 (6.00)	139.15 (5.00)	163.40 (6.50)	130.03 (4.50)	120.10 (5.00)	139.13 (5.50)	154.87 (5.50)	146.20 (5.50)
	Log(C_{R2})	98.52 (4.50)	69.73 (3.00)	81.70 (3.50)	89.87 (3.00)	106.72 (4.50)	**44.57 (1.50)**	46.23 (2.00)	56.37 (2.00)	92.17 (3.50)
	Metric(CD)	141.42 (4.50)	164.30 (6.00)	159.33 (5.50)	159.63 (6.50)	123.45 (4.50)	137.37 (5.00)	143.00 (5.50)	164.00 (5.50)	145.13 (5.50)
	Metric(C_{R2})	**18.68 (1.00)**	**21.10 (1.00)**	**16.67 (1.00)**	68.63 (3.00)	**15.50 (1.00)**	134.07 (5.00)	**26.10 (1.50)**	**17.27 (1.50)**	**21.47 (1.00)**

Considering the ranking strategies, in all cases the TWFs based on the C_{R2} indicator outperformed other algorithms more times than those based on the CD indicator. In a more detailed view we can see that for more than five objectives, in the few times a TWF based on CD is among the best, it is statistically tied with another one based on C_{R2}, which indicates that CD does not scale well with the number of objectives.

Comparing the weighting functions, we can identify a pattern where the number of times they outperform others (using the same ranking strategy) can be expressed as linear < log < metric. Again in more detail, this trend becomes more evident as the number of objectives increases.

In summary, among the compared TWFs, metric(C_{R2}) is the most indicated for many-objective instances of the DTLZ problems.

Table 7.7: Summarized table of the IGD_p results obtained by each MO-CMA-ES variant for the DTLZ problem up to 20 objectives. The upper part shows the number of times the algorithm outperformed the others. The lower part shows the sum of the times an algorithm outperformed all the others.

Alg.	1	2	3	4	5	6	7
1	n/a	5	4	6	13	9	22
2	3	n/a	7	2	15	3	24
3	1	1	n/a	3	6	4	19
4	4	0	8	n/a	16	1	18
5	1	1	0	1	n/a	1	15
6	5	0	5	0	12	n/a	19
7	5	7	10	4	10	6	n/a

Alg. #	Outperformed the others	Title
1	19	Equal
2	14	Linear(CD)
3	34	Linear(C_{R2})
4	16	Log(CD)
5	72	Log(C_{R2})
6	24	Metric(CD)
7	117	Metric(C_{R2})

Table 7.8: Summarized table of the Hypervolume results obtained by each MO-CMA-ES variant for the DTLZ problem up to 20 objectives. The upper part shows the number of times the algorithm outperformed the others. The lower part shows the sum of the times an algorithm outperformed all the others.

Alg.	1	2	3	4	5	6	7
1	n/a	0	8	2	11	4	18
2	4	n/a	6	0	17	2	20
3	1	0	n/a	0	6	0	14
4	5	1	7	n/a	17	1	19
5	2	2	2	2	n/a	3	10
6	4	0	8	1	17	n/a	21
7	11	9	15	8	12	9	n/a

Alg. #	Outperformed the others	Title
1	27	Equal
2	12	Linear(CD)
3	46	Linear(C_{R2})
4	13	Log(CD)
5	80	Log(C_{R2})
6	19	Metric(CD)
7	102	Metric(C_{R2})

Next we present our considerations with respect to the results obtained for the WFG problems. From the summarized IGD_p table, we can appreciate that the equal TWF outperformed the other algorithms more times, followed by linear(C_{R2}) and log(C_{R2}). Considering the HV, this trend is reversed, and all the TWFs using C_{R2} as ranking method outperformed the other

algorithms more times than equal. Another trend identified is that the weighting strategies that introduce less selection pressure in the search present good results when considering a low number of objectives, but when this number increases their performance deteriorate. This can be seen in all Tables (7.3, 7.4, 7.5 and 7.6) where equal and linear appear among the best algorithms more times for three and five objectives, while metric(C_{R2}) is usually among the best for fifteen and twenty objectives.

Table 7.9: Summarized table of the IGD_p results obtained by each MO-CMA-ES variant for the WFG problem up to 20 objectives. The upper part shows the number of times the algorithm outperformed the others. The lower part shows the sum of the times an algorithm outperformed all the others.

Alg.	1	2	3	4	5	6	7
1	n/a	1	5	6	10	3	5
2	9	n/a	6	2	6	0	6
3	7	0	n/a	0	1	0	6
4	22	6	14	n/a	10	1	9
5	13	9	7	1	n/a	1	7
6	23	10	19	1	10	n/a	11
7	33	31	33	28	29	23	n/a

Alg. #	Outperformed the others	Title
1	107	Equal
2	57	Linear(CD)
3	84	Linear(C_{R2})
4	38	Log(CD)
5	66	Log(C_{R2})
6	28	Metric(CD)
7	44	Metric(C_{R2})

Table 7.10: Summarized table of the Hypervolume results obtained by each MO-CMA-ES variant for the WFG problem up to 20 objectives. The upper part shows the number of times the algorithm outperformed the others. The lower part shows the sum of the times an algorithm outperformed all the others.

Alg.	1	2	3	4	5	6	7
1	n/a	0	8	0	9	0	17
2	13	n/a	18	0	16	0	21
3	3	0	n/a	0	1	0	19
4	29	11	38	n/a	33	1	27
5	9	1	6	0	n/a	0	16
6	29	15	39	1	36	n/a	28
7	25	20	24	12	23	11	n/a

Alg. #	Outperformed the others	Title
1	108	Equal
2	47	Linear(CD)
3	133	Linear(C_{R2})
4	13	Log(CD)
5	118	Log(C_{R2})
6	12	Metric(CD)
7	128	Metric(C_{R2})

7.4.1.3 Comparison between the algorithms up to 8 objectives

For eight or fewer objectives, it is possible to calculate exactly C_{HV} as a metric for ranking the algorithms in a reasonable time, hence, we conducted experiments to compare the behavior of this rank strategy associated with the three weighting strategies that consider the ranking (linear, log and metric) to the previous results for 3, 5 and 8 objectives. We hypothesize that by using a powerful indicator such as C_{HV}, the results of the weighting functions that introduce more selection pressure in the model will improve, following the trend seen in the last section, where C_{R2}, in general, performed better than CD.

Tables 7.11 and 7.12 present the extended comparison results measured by the IGD_p for the DTLZ and WFG problems respectively. The results measured by HV are presented in Tables 7.13 and 7.14 for DTLZ and WFG respectively. The summarized results for the DTLZ family are presented in Tables 7.15 and 7.16 for the IGD_p and HV. For the WFG family the summarized results are presented in Tables 7.17 and 7.18.

First, we introduce the results obtained for the DTLZ problems, where we can observe in the tables that equal never outperformed other TWFs more times than any strategy using C_{HV}. Moreover, equal is seldom among the best algorithms, just a total of 8 times considering both indicators, from which on two cases all the algorithms tied, so it cannot be considered "better" than any other.

Comparing the algorithms in terms of ranking strategy, the number of times the TWFs outperformed others can be expressed as CD $< C_{R2} < C_{HV}$ in both tables, which indicates that the use of a more powerful metric had a good impact on the results. This trend becomes more evident as the number of objectives increases.

When considering the weighting functions, the results obtained using the IGD_p indicator present clear patterns, where, in general, it is best to use the metric approach, especially in the many-objective scenario. If we consider the results measured by the HV indicator, the patterns regarding the weighting function are less clear, possibly due to the higher influence of the C_{HV} as ranking method, where in many cases different weighting functions using this indicator are statistically tied.

Table 7.11: Mean ranks of the IGD_p as used in the Kruskal-Wallis test for the DTLZ problems up to 8 objectives. Final ranks, presented in parentheses assigned according to mean ranks.

Obj.	Algorithms	DTLZ1	DTLZ2	DTLZ3	DTLZ4	DTLZ5	DTLZ6	DTLZ7
3	Equal	171.77 (5.50)	176.93 (6.00)	**115.00 (5.00)**	**134.43 (5.50)**	**85.67 (3.50)**	210.83 (8.00)	183.10 (7.00)
	Linear(CD)	129.47 (5.50)	147.50 (5.00)	134.63 (5.50)	**179.68 (5.50)**	109.40 (4.00)	178.47 (7.50)	109.67 (4.50)
	Linear(C_{R2})	**114.60 (5.00)**	110.83 (5.00)	140.57 (5.50)	**126.30 (5.50)**	97.50 (4.00)	**106.73 (3.00)**	161.87 (6.00)
	Linear(C_{HV})	145.60 (5.50)	132.83 (5.00)	166.53 (5.50)	**155.97 (5.50)**	161.17 (5.00)	114.70 (4.00)	172.73 (6.50)
	Log(CD)	146.30 (5.50)	165.63 (5.00)	156.03 (5.50)	**170.28 (5.50)**	103.50 (4.00)	207.67 (8.00)	**75.50 (2.00)**
	Log(C_{R2})	143.07 (5.50)	115.83 (5.00)	142.20 (5.50)	**120.40 (5.50)**	90.67 (4.00)	**98.37 (3.00)**	159.27 (6.00)
	Log(C_{HV})	141.17 (5.50)	**102.37 (4.50)**	148.97 (5.50)	**144.30 (5.50)**	185.40 (7.50)	**88.80 (3.00)**	173.13 (6.50)
	Metric(CD)	144.50 (5.50)	163.13 (5.00)	141.90 (5.50)	**129.83 (5.50)**	131.77 (4.50)	180.20 (7.50)	95.23 (3.00)
	Metric(C_{R2})	187.27 (6.00)	285.50 (10.00)	165.30 (5.50)	**184.18 (5.50)**	285.50 (9.50)	248.93 (8.00)	166.13 (6.50)
	Metric(C_{HV})	181.27 (5.50)	**104.43 (4.50)**	193.87 (6.00)	**159.62 (5.50)**	254.43 (9.00)	**70.30 (3.00)**	208.37 (7.00)
5	Equal	161.77 (5.50)	189.87 (7.00)	124.53 (4.50)	145.20 (5.50)	227.70 (8.00)	200.67 (7.00)	143.97 (5.00)
	Linear(CD)	180.20 (6.50)	220.93 (7.00)	188.73 (6.50)	129.60 (5.00)	187.33 (6.50)	225.57 (7.50)	75.40 (3.50)
	Linear(C_{R2})	157.60 (5.50)	186.30 (7.00)	197.70 (7.00)	**98.67 (4.00)**	178.87 (6.50)	193.37 (7.00)	131.00 (4.50)
	Linear(C_{HV})	148.77 (5.50)	**76.73 (2.00)**	143.00 (5.50)	145.63 (5.50)	142.43 (5.50)	97.87 (2.50)	206.13 (8.00)
	Log(CD)	161.90 (5.50)	224.17 (7.00)	173.40 (6.50)	195.83 (6.50)	193.13 (6.50)	189.67 (7.00)	97.93 (4.00)
	Log(C_{R2})	138.83 (5.50)	173.37 (7.00)	203.40 (7.00)	122.97 (4.50)	137.40 (5.50)	190.30 (7.00)	128.63 (4.50)
	Log(C_{HV})	188.17 (6.50)	**43.00 (2.00)**	142.40 (5.50)	145.67 (5.50)	73.17 (2.50)	**36.57 (2.00)**	255.80 (9.00)
	Metric(CD)	181.20 (6.50)	212.57 (7.00)	135.90 (5.50)	145.30 (5.50)	150.87 (6.00)	196.33 (7.00)	**50.67 (2.00)**
	Metric(C_{R2})	**94.03 (4.00)**	155.03 (7.00)	**96.13 (3.50)**	202.80 (7.00)	191.73 (6.50)	145.03 (6.00)	155.80 (5.50)
	Metric(C_{HV})	**92.53 (4.00)**	**23.03 (2.00)**	**99.80 (3.50)**	173.33 (6.00)	**22.37 (1.50)**	**29.63 (2.00)**	259.67 (9.00)
8	Equal	196.93 (7.00)	210.30 (7.50)	205.57 (7.00)	**107.23 (3.50)**	233.17 (8.50)	188.17 (7.00)	98.80 (4.00)
	Linear(CD)	207.10 (7.00)	227.20 (8.00)	203.03 (7.00)	152.60 (5.50)	246.23 (8.50)	228.50 (7.50)	151.77 (6.00)
	Linear(C_{R2})	182.80 (6.50)	188.17 (7.00)	172.67 (7.00)	113.93 (4.50)	193.87 (7.00)	170.17 (7.00)	**64.53 (2.50)**
	Linear(C_{HV})	138.87 (6.50)	133.73 (4.50)	141.20 (6.00)	108.30 (4.00)	108.97 (4.00)	185.83 (7.00)	**71.03 (2.50)**
	Log(CD)	196.50 (7.00)	231.00 (8.00)	196.27 (7.00)	188.67 (7.00)	198.67 (7.00)	211.90 (7.00)	181.27 (6.50)
	Log(C_{R2})	164.23 (6.50)	143.23 (5.00)	187.77 (7.00)	120.70 (5.00)	135.27 (5.50)	144.37 (6.00)	**73.37 (2.50)**
	Log(C_{HV})	125.57 (4.50)	**50.20 (2.00)**	83.30 (2.50)	132.33 (5.00)	45.00 (2.00)	**72.73 (2.00)**	270.80 (9.50)
	Metric(CD)	202.20 (7.00)	221.77 (8.00)	219.90 (7.50)	181.20 (6.50)	159.90 (5.50)	208.63 (7.00)	165.10 (6.00)
	Metric(C_{R2})	**47.93 (1.50)**	81.37 (3.00)	**53.07 (2.00)**	220.93 (8.00)	155.20 (5.50)	74.13 (2.50)	163.07 (6.00)
	Metric(C_{HV})	**42.87 (1.50)**	**18.03 (2.00)**	**42.23 (2.00)**	179.10 (6.00)	**28.73 (1.50)**	**20.57 (2.00)**	265.27 (9.50)

Table 7.12: Mean ranks of the IGD_p as used in the Kruskal-Wallis test for the WFG problems up to 8 objectives. Final ranks, presented in parentheses assigned according to mean ranks.

Obj.	Algorithms	WFG1	WFG2	WFG3	WFG4	WFG5	WFG6	WFG7	WFG8	WFG9
3	Equal	**129.47 (4.00)**	91.43 (3.50)	141.33 (5.00)	176.83 (6.00)	110.27 (4.00)	113.90 (4.00)	130.40 (5.50)	130.97 (5.50)	163.93 (5.50)
	Linear(CD)	**141.20 (4.00)**	96.93 (3.50)	173.90 (6.50)	147.90 (5.50)	169.23 (6.50)	150.80 (5.50)	143.57 (5.50)	152.90 (5.50)	147.07 (5.50)
	Linear(C_{R2})	**91.33 (4.00)**	90.03 (3.50)	112.43 (4.00)	132.13 (5.00)	86.33 (3.00)	101.70 (4.00)	115.83 (4.50)	104.87 (3.50)	156.07 (5.50)
	Linear(C_{HV})	**93.67 (4.00)**	**72.57 (3.00)**	72.13 (3.00)	86.10 (3.00)	**50.17 (2.50)**	**59.33 (2.50)**	**32.27 (1.50)**	**43.03 (2.00)**	**107.03 (4.50)**
	Log(CD)	**143.07 (4.00)**	146.73 (5.00)	189.53 (7.00)	157.57 (6.00)	216.47 (8.00)	195.47 (7.00)	193.10 (6.50)	201.10 (6.50)	152.17 (5.50)
	Log(C_{R2})	**97.83 (4.00)**	132.70 (4.50)	99.93 (3.50)	126.27 (5.00)	136.60 (4.50)	145.57 (5.50)	152.10 (5.50)	130.60 (5.50)	147.00 (5.50)
	Log(C_{HV})	**88.33 (4.00)**	174.93 (7.00)	**54.43 (2.50)**	**66.77 (2.50)**	71.90 (3.00)	**57.50 (2.50)**	46.17 (2.00)	**45.37 (2.00)**	118.67 (5.00)
	Metric(CD)	273.90 (9.00)	198.70 (7.50)	220.27 (8.00)	162.87 (6.00)	174.70 (6.50)	163.33 (5.50)	160.20 (5.50)	184.27 (6.50)	186.90 (6.00)
	Metric(C_{R2})	226.23 (9.00)	244.53 (8.50)	270.60 (9.50)	281.33 (10.00)	234.13 (8.00)	283.07 (9.50)	282.83 (9.50)	284.17 (9.50)	198.37 (6.50)
	Metric(C_{HV})	219.97 (9.00)	256.43 (9.00)	170.43 (6.00)	167.23 (6.00)	255.20 (9.00)	234.33 (9.00)	248.53 (9.00)	227.73 (8.50)	127.80 (5.50)
5	Equal	131.07 (4.50)	148.43 (5.50)	244.60 (8.00)	127.73 (4.50)	131.10 (5.00)	151.87 (5.50)	165.50 (6.00)	95.03 (4.00)	152.10 (6.50)
	Linear(CD)	154.93 (6.00)	179.53 (6.00)	207.07 (7.50)	134.27 (4.50)	161.67 (6.00)	146.97 (5.50)	161.93 (6.00)	160.17 (6.00)	178.07 (7.00)
	Linear(C_{R2})	137.70 (4.50)	**63.37 (2.50)**	151.30 (5.50)	**101.17 (4.00)**	152.63 (6.00)	116.10 (4.50)	131.33 (5.50)	97.97 (4.00)	166.63 (7.00)
	Linear(C_{HV})	106.60 (4.00)	69.67 (3.00)	63.60 (2.50)	**95.37 (4.00)**	**38.90 (1.50)**	**52.07 (2.00)**	**50.47 (1.50)**	**42.67 (2.50)**	**76.27 (2.00)**
	Log(CD)	195.03 (7.00)	134.57 (5.00)	194.63 (7.00)	153.93 (4.50)	213.43 (7.00)	165.23 (5.50)	190.67 (6.00)	225.07 (7.50)	188.07 (7.00)
	Log(C_{R2})	167.27 (6.50)	111.63 (4.50)	82.80 (3.00)	135.07 (4.50)	168.57 (6.50)	151.10 (5.50)	124.33 (5.50)	155.87 (6.00)	172.23 (7.00)
	Log(C_{HV})	**77.37 (3.00)**	175.20 (6.00)	**23.30 (2.00)**	**98.03 (4.00)**	92.87 (3.50)	**47.47 (2.00)**	65.03 (2.50)	**66.83 (2.50)**	86.37 (2.50)
	Metric(CD)	237.90 (8.50)	127.97 (4.50)	223.10 (8.00)	173.77 (6.50)	204.00 (7.00)	167.30 (6.00)	157.53 (6.00)	162.03 (6.00)	209.70 (7.00)
	Metric(C_{R2})	217.43 (8.00)	235.67 (8.50)	190.17 (7.00)	254.87 (9.50)	189.73 (6.50)	269.87 (9.50)	269.07 (10.00)	277.67 (9.00)	209.97 (7.00)
	Metric(C_{HV})	**79.70 (3.00)**	258.97 (9.50)	124.43 (4.50)	230.80 (9.00)	152.10 (6.00)	237.03 (9.00)	189.13 (6.00)	221.70 (7.50)	**65.60 (2.00)**
8	Equal	**101.73 (4.00)**	198.17 (7.50)	218.67 (7.50)	**73.23 (2.50)**	76.30 (2.00)	**59.53 (2.50)**	161.47 (5.50)	66.63 (2.50)	132.27 (5.00)
	Linear(CD)	177.37 (6.00)	185.60 (7.50)	201.70 (7.00)	**64.67 (2.50)**	166.20 (6.50)	**69.97 (2.50)**	98.13 (4.00)	136.20 (5.00)	141.77 (5.00)
	Linear(C_{R2})	139.63 (5.00)	110.60 (3.00)	170.10 (6.50)	**58.63 (2.50)**	159.60 (6.50)	**53.73 (2.50)**	**89.83 (3.50)**	178.77 (7.00)	155.63 (5.00)
	Linear(C_{HV})	121.97 (5.00)	**69.73 (2.50)**	66.37 (2.50)	**81.87 (2.50)**	**70.93 (1.50)**	89.37 (3.00)	**85.67 (3.50)**	**50.23 (2.00)**	**88.70 (4.00)**
	Log(CD)	183.87 (6.50)	170.30 (6.50)	136.13 (5.00)	198.00 (7.50)	196.00 (6.50)	160.20 (6.00)	99.97 (4.00)	160.37 (6.00)	182.07 (6.50)
	Log(C_{R2})	165.63 (5.50)	102.10 (3.00)	130.43 (4.50)	197.27 (7.50)	169.57 (6.50)	184.60 (6.50)	92.20 (4.00)	194.97 (7.00)	144.80 (5.00)
	Log(C_{HV})	109.67 (4.50)	**73.17 (2.50)**	**17.67 (1.50)**	188.53 (7.50)	144.03 (6.00)	198.57 (7.50)	229.67 (8.50)	79.50 (2.50)	**94.07 (4.00)**
	Metric(CD)	222.90 (8.00)	201.80 (7.50)	158.27 (6.50)	212.13 (7.50)	196.00 (6.50)	168.50 (6.50)	116.20 (4.00)	169.77 (7.00)	172.13 (6.00)
	Metric(C_{R2})	160.00 (5.50)	200.30 (7.50)	188.87 (6.50)	190.17 (7.50)	168.93 (6.50)	256.70 (9.00)	256.97 (9.00)	233.57 (8.00)	248.27 (9.50)
	Metric(C_{HV})	122.23 (5.00)	193.23 (7.50)	216.80 (7.50)	240.50 (7.50)	157.43 (6.50)	263.83 (9.00)	274.90 (9.00)	235.00 (8.00)	145.30 (5.00)

Table 7.13: Mean ranks of the Hypervolume as used in the Kruskal-Wallis test for the DTLZ problems up to 8 objectives. Final ranks, presented in parentheses assigned according to mean ranks.

Obj.	Algorithms	DTLZ1	DTLZ2	DTLZ3	DTLZ4	DTLZ5	DTLZ6	DTLZ7
3	Equal	110.87 (4.00)	176.57 (6.50)	**62.10 (3.00)**	**141.90 (5.50)**	160.53 (5.00)	203.90 (8.00)	163.53 (5.50)
	Linear(CD)	**73.77 (3.50)**	154.73 (6.00)	92.40 (3.50)	**167.78 (5.50)**	130.00 (4.50)	166.93 (5.50)	135.43 (5.00)
	Linear(C_{R2})	**92.37 (3.50)**	108.60 (4.00)	107.10 (3.50)	**127.80 (5.50)**	106.93 (4.00)	99.80 (3.50)	117.43 (5.00)
	Linear(C_{HV})	138.13 (4.50)	93.83 (3.50)	136.83 (4.50)	**166.03 (5.50)**	91.13 (4.00)	111.67 (4.00)	**95.90 (3.50)**
	Log(CD)	116.67 (4.00)	199.33 (7.00)	119.57 (4.00)	**164.45 (5.50)**	133.97 (4.50)	206.17 (8.00)	168.37 (6.50)
	Log(C_{R2})	171.40 (5.50)	157.80 (6.00)	181.17 (7.00)	**108.23 (5.50)**	**74.03 (3.50)**	104.90 (3.50)	159.30 (5.50)
	Log(C_{HV})	190.03 (7.50)	**67.43 (2.50)**	208.50 (8.50)	**162.73 (5.50)**	107.57 (4.50)	104.33 (3.50)	**96.60 (3.50)**
	Metric(CD)	101.03 (4.00)	182.27 (7.00)	111.53 (4.00)	**135.37 (5.50)**	177.90 (6.50)	175.87 (7.00)	171.53 (6.50)
	Metric(C_{R2})	262.80 (9.50)	285.50 (10.00)	239.50 (8.50)	**156.82 (5.50)**	285.50 (9.50)	262.03 (9.00)	209.53 (7.50)
	Metric(C_{HV})	247.93 (9.00)	**78.93 (2.50)**	246.30 (8.50)	**173.88 (5.50)**	237.43 (9.00)	**69.40 (3.00)**	187.37 (6.50)
5	Equal	106.37 (4.50)	192.37 (7.00)	**75.00 (3.50)**	130.93 (5.00)	245.37 (8.50)	201.37 (7.00)	123.13 (4.00)
	Linear(CD)	113.10 (4.50)	217.37 (7.00)	149.37 (5.00)	136.60 (5.00)	208.93 (7.50)	232.97 (7.50)	232.07 (8.50)
	Linear(C_{R2})	125.37 (4.50)	178.47 (7.00)	145.10 (4.50)	**91.30 (4.00)**	156.73 (6.00)	189.43 (7.00)	92.17 (3.50)
	Linear(C_{HV})	123.00 (4.50)	**72.77 (2.00)**	99.40 (4.00)	151.83 (5.50)	82.63 (2.50)	**85.90 (2.00)**	**52.53 (3.00)**
	Log(CD)	**80.07 (3.50)**	199.00 (7.00)	118.67 (4.00)	208.53 (8.00)	198.80 (7.50)	190.93 (7.00)	270.40 (9.00)
	Log(C_{R2})	162.17 (5.00)	158.00 (7.00)	190.93 (7.50)	**110.83 (4.00)**	124.10 (4.00)	177.37 (7.00)	94.80 (3.50)
	Log(C_{HV})	168.00 (5.00)	**40.17 (2.00)**	106.80 (4.00)	155.87 (5.50)	**35.90 (2.00)**	**33.17 (2.00)**	**73.60 (3.00)**
	Metric(CD)	105.83 (4.50)	200.53 (7.00)	103.17 (4.00)	133.50 (5.00)	157.93 (6.50)	203.17 (7.00)	241.47 (9.00)
	Metric(C_{R2})	274.17 (9.50)	213.57 (7.00)	248.33 (9.00)	196.83 (6.50)	228.73 (8.00)	161.20 (6.50)	167.37 (6.50)
	Metric(C_{HV})	246.93 (9.50)	**32.77 (2.00)**	268.23 (9.50)	188.77 (6.50)	65.87 (2.50)	**29.50 (2.00)**	157.47 (5.00)
8	Equal	155.93 (6.00)	205.37 (7.00)	175.87 (6.50)	**108.20 (4.00)**	243.80 (8.00)	185.13 (7.00)	222.07 (8.00)
	Linear(CD)	188.60 (6.50)	240.90 (8.00)	200.47 (6.50)	152.13 (5.00)	250.50 (8.00)	229.40 (8.00)	221.07 (8.00)
	Linear(C_{R2})	133.67 (4.50)	187.37 (7.00)	152.47 (6.50)	**106.90 (4.00)**	184.30 (7.50)	170.70 (6.50)	202.37 (7.50)
	Linear(C_{HV})	**81.57 (3.00)**	98.37 (3.50)	**75.77 (1.50)**	**106.50 (4.00)**	86.33 (3.00)	145.20 (5.00)	111.03 (4.00)
	Log(CD)	140.67 (4.50)	210.37 (7.50)	195.43 (6.50)	186.57 (7.00)	221.10 (8.00)	219.00 (8.00)	231.87 (8.00)
	Log(C_{R2})	136.87 (4.50)	143.60 (6.00)	149.83 (6.50)	118.87 (5.00)	108.70 (3.50)	141.10 (5.00)	146.47 (4.50)
	Log(C_{HV})	**71.93 (3.00)**	41.87 (2.00)	**30.43 (1.50)**	131.57 (5.00)	**17.33 (2.00)**	65.10 (2.00)	**29.87 (2.00)**
	Metric(CD)	215.27 (8.00)	213.83 (7.50)	207.17 (6.50)	166.60 (5.00)	194.70 (7.50)	219.77 (8.00)	225.73 (8.00)
	Metric(C_{R2})	227.97 (9.00)	138.30 (5.00)	161.37 (6.50)	239.60 (9.00)	139.77 (5.00)	110.93 (4.00)	77.80 (3.00)
	Metric(C_{HV})	152.53 (6.00)	**25.03 (1.50)**	156.20 (6.50)	188.07 (7.00)	58.47 (2.50)	**18.67 (1.50)**	**36.73 (2.00)**

Table 7.14: Mean ranks of the Hypervolume as used in the Kruskal-Wallis test for the WFG problems up to 8 objectives. Final ranks, presented in parentheses assigned according to mean ranks.

Obj.	Algorithms	WFG1	WFG2	WFG3	WFG4	WFG5	WFG6	WFG7	WFG8	WFG9
3	Equal	**126.33 (4.00)**	75.27 (3.00)	147.27 (6.00)	185.23 (6.00)	113.67 (4.00)	114.67 (4.50)	**117.83 (5.50)**	130.90 (5.00)	151.83 (5.50)
	Linear(CD)	**133.47 (4.00)**	102.03 (4.00)	159.93 (6.00)	158.93 (6.00)	171.63 (6.50)	156.77 (5.50)	**141.83 (5.50)**	153.23 (5.50)	156.93 (5.50)
	Linear(C_{R2})	**86.37 (4.00)**	74.70 (3.00)	115.07 (4.00)	134.07 (5.50)	86.00 (3.00)	108.03 (4.00)	**110.53 (5.50)**	105.50 (3.50)	152.03 (5.50)
	Linear(C_{HV})	**98.83 (4.00)**	**61.53 (2.50)**	61.87 (2.50)	79.77 (3.00)	**45.03 (2.50)**	46.03 (2.50)	**26.60 (5.50)**	**35.13 (2.00)**	**93.67 (4.50)**
	Log(CD)	**145.33 (4.00)**	157.13 (6.00)	192.47 (7.00)	172.60 (6.00)	217.20 (8.00)	203.97 (7.00)	**198.43 (5.50)**	203.63 (7.00)	161.83 (5.50)
	Log(C_{R2})	**96.50 (4.00)**	142.17 (5.00)	116.07 (4.00)	128.87 (5.50)	128.93 (4.50)	153.77 (5.50)	**157.10 (5.50)**	134.67 (5.50)	148.00 (5.50)
	Log(C_{HV})	**82.23 (4.00)**	168.90 (6.00)	**28.27 (1.50)**	**47.13 (1.50)**	69.77 (3.00)	**42.27 (2.00)**	**49.93 (5.50)**	**42.20 (2.00)**	**117.90 (4.50)**
	Metric(CD)	275.40 (9.00)	205.17 (7.50)	216.53 (7.50)	169.50 (6.00)	173.93 (6.50)	165.53 (6.00)	**166.77 (5.50)**	181.93 (6.50)	189.43 (6.50)
	Metric(C_{R2})	235.10 (9.00)	253.23 (9.00)	275.70 (9.50)	284.23 (10.00)	241.40 (8.00)	284.37 (9.50)	**284.67 (5.50)**	284.23 (9.50)	204.37 (7.00)
	Metric(C_{HV})	225.43 (9.00)	264.87 (9.00)	191.83 (7.00)	144.67 (5.50)	257.43 (9.00)	229.60 (8.50)	**251.30 (5.50)**	233.57 (8.50)	129.00 (5.00)
5	Equal	101.70 (4.00)	117.83 (4.00)	199.37 (7.50)	160.73 (6.00)	153.10 (5.50)	169.43 (6.00)	**174.27 (5.50)**	**151.47 (5.50)**	144.37 (6.00)
	Linear(CD)	147.47 (5.00)	154.03 (5.50)	169.27 (6.50)	157.30 (6.00)	184.60 (7.00)	161.90 (6.00)	**171.70 (5.50)**	**189.27 (5.50)**	203.33 (7.00)
	Linear(C_{R2})	114.50 (4.00)	**60.37 (2.50)**	112.00 (3.50)	101.00 (4.00)	188.83 (7.50)	117.90 (5.50)	**115.10 (5.50)**	**113.33 (5.50)**	184.13 (7.00)
	Linear(C_{HV})	**62.53 (2.50)**	**55.80 (2.50)**	50.67 (2.50)	**71.83 (2.50)**	**37.90 (2.00)**	**46.50 (1.50)**	**29.70 (5.50)**	**40.40 (5.50)**	**65.57 (2.00)**
	Log(CD)	219.67 (9.00)	176.93 (6.00)	189.43 (7.00)	173.77 (6.50)	226.27 (8.00)	167.07 (6.00)	**196.33 (5.50)**	**237.80 (5.50)**	218.20 (7.50)
	Log(C_{R2})	146.50 (5.00)	92.50 (3.50)	82.30 (2.50)	134.33 (5.00)	184.67 (7.00)	128.53 (5.50)	**136.80 (5.50)**	**109.03 (5.50)**	169.40 (7.00)
	Log(C_{HV})	**62.93 (2.50)**	136.40 (5.50)	**16.80 (2.00)**	**68.23 (2.50)**	92.13 (3.00)	**32.43 (1.50)**	**36.17 (5.50)**	**21.03 (5.50)**	**59.27 (2.00)**
	Metric(CD)	270.03 (9.00)	200.43 (7.50)	233.17 (7.50)	170.37 (6.00)	216.07 (7.50)	181.73 (6.00)	**167.73 (5.50)**	**180.43 (5.50)**	224.50 (7.50)
	Metric(C_{R2})	243.57 (9.00)	248.60 (9.00)	266.97 (9.00)	268.03 (9.50)	117.77 (4.50)	282.03 (9.50)	**283.60 (5.50)**	**283.03 (5.50)**	181.80 (7.00)
	Metric(C_{HV})	136.10 (5.00)	262.10 (9.00)	185.03 (7.00)	199.40 (7.00)	103.67 (3.00)	217.47 (7.50)	**193.60 (5.50)**	**179.20 (5.50)**	**54.43 (2.00)**
8	Equal	**52.27 (3.00)**	125.20 (4.50)	196.90 (7.50)	116.80 (4.50)	150.77 (6.00)	112.20 (4.50)	136.30 (5.00)	117.53 (4.50)	192.40 (7.50)
	Linear(CD)	211.97 (8.00)	178.40 (6.50)	202.63 (7.50)	116.13 (4.50)	219.67 (7.50)	150.70 (5.00)	155.30 (6.00)	201.60 (7.50)	228.97 (7.50)
	Linear(C_{R2})	111.23 (3.50)	92.70 (3.00)	139.80 (5.00)	93.13 (3.50)	178.40 (6.50)	93.77 (3.00)	121.90 (5.00)	189.97 (7.50)	177.80 (7.00)
	Linear(C_{HV})	95.63 (3.50)	**51.03 (2.50)**	46.67 (2.50)	**39.30 (2.00)**	105.97 (3.50)	**29.80 (2.00)**	**36.17 (1.50)**	**39.43 (2.00)**	**85.80 (2.50)**
	Log(CD)	223.43 (8.50)	201.23 (8.00)	229.23 (8.00)	221.00 (8.00)	197.27 (7.50)	231.53 (8.00)	212.50 (7.50)	254.57 (8.50)	256.03 (8.50)
	Log(C_{R2})	192.33 (8.00)	115.40 (4.00)	114.67 (3.50)	161.50 (6.00)	169.47 (6.00)	175.20 (6.50)	211.30 (7.50)	173.83 (6.50)	159.20 (6.50)
	Log(C_{HV})	101.57 (3.50)	114.80 (4.00)	**16.07 (1.50)**	81.23 (3.00)	114.87 (4.00)	52.77 (2.50)	**35.17 (1.50)**	**21.57 (2.00)**	**34.30 (2.50)**
	Metric(CD)	249.73 (8.50)	244.20 (8.00)	253.13 (8.00)	263.67 (9.00)	198.87 (7.50)	235.97 (8.00)	180.97 (6.50)	234.07 (7.50)	228.83 (7.50)
	Metric(C_{R2})	152.10 (5.00)	196.93 (8.00)	214.10 (8.00)	227.60 (8.00)	**56.60 (2.50)**	257.30 (9.00)	250.27 (8.50)	183.10 (6.50)	101.63 (3.00)
	Metric(C_{HV})	114.73 (3.50)	185.10 (6.50)	91.80 (3.50)	184.63 (6.50)	113.13 (4.00)	165.77 (6.50)	165.13 (6.00)	89.33 (2.50)	**40.03 (2.50)**

Table 7.15: Summarized table of the IGD_p results obtained by each MO-CMA-ES variant for the DTLZ problem up to 8 objectives. The upper part shows the number of times the algorithm outperformed the others. The lower part shows the sum of the times an algorithm outperformed all the others.

Alg.	1	2	3	4	5	6	7	8	9	10
1	n/a	1	1	6	1	3	10	4	5	10
2	0	n/a	2	5	0	5	9	1	8	11
3	1	0	n/a	3	1	0	7	1	5	9
4	1	1	1	n/a	2	1	2	2	3	6
5	2	0	4	7	n/a	4	9	0	5	10
6	1	0	0	2	1	n/a	6	1	3	9
7	3	3	3	1	4	3	n/a	3	4	2
8	1	0	2	6	0	3	9	n/a	5	10
9	3	4	7	6	3	6	8	4	n/a	6
10	5	4	4	2	4	3	0	4	2	n/a

Alg. #	Outperformed the others	Title
1	17	Equal
2	13	Linear(CD)
3	24	Linear(C_{R2})
4	38	Linear(C_{HV})
5	16	Log(CD)
6	28	Log(C_{R2})
7	60	Log(C_{HV})
8	20	Metric(CD)
9	40	Metric(C_{R2})
10	73	Metric(C_{HV})

Table 7.16: Summarized table of the Hypervolume results obtained by each MO-CMA-ES variant for the DTLZ problem up to 8 objectives. The upper part shows the number of times the algorithm outperformed the others. The lower part shows the sum of the times an algorithm outperformed all the others.

Alg.	1	2	3	4	5	6	7	8	9	10
1	n/a	0	2	10	0	5	11	1	3	9
2	2	n/a	1	10	0	6	11	0	5	10
3	0	0	n/a	7	0	1	8	0	1	7
4	1	0	0	n/a	0	0	2	0	0	3
5	3	1	5	13	n/a	7	12	1	5	10
6	2	2	2	4	2	n/a	10	1	0	6
7	2	2	2	1	3	0	n/a	2	0	0
8	1	0	5	13	1	7	13	n/a	4	10
9	8	9	13	15	7	10	16	8	n/a	9
10	6	5	7	10	5	5	7	4	0	n/a

Alg. #	Outperformed the others	Title
1	25	Equal
2	19	Linear(CD)
3	37	Linear(C_{R2})
4	83	Linear(C_{HV})
5	18	Log(CD)
6	41	Log(C_{R2})
7	90	Log(C_{HV})
8	17	Metric(CD)
9	18	Metric(C_{R2})
10	64	Metric(C_{HV})

Next, we present the results obtained for the WFG problems, where both summarized tables (7.17 and 7.18) show a better performance for the TWFs linear(C_{HV}) and log(C_{HV}) over the others for these problems. The same trend can be seen in the general tables, where for all the numbers of objectives, these two TWFs are highlighted most of the times. These results indicate that using a powerful metric to rank the solutions can, in fact, improve the performance of MO-CMA-ES. Regarding the weighting strategies, the number of times any of them outperformed the other TWFs can be expressed as linear > log > metric considering both indicators.

Table 7.17: Summarized table of the IGD_p results obtained by each MO-CMA-ES variant for the WFG problem up to 8 objectives. The upper part shows the number of times the algorithm outperformed the others. The lower part shows the sum of the times an algorithm outperformed all the others.

Alg.	1	2	3	4	5	6	7	8	9	10
1	n/a	0	4	12	1	3	9	0	0	2
2	2	n/a	3	16	0	4	14	0	0	3
3	2	0	n/a	8	0	0	4	0	0	1
4	0	0	0	n/a	0	0	0	0	0	0
5	9	2	9	19	n/a	3	18	0	0	2
6	4	2	2	13	0	n/a	10	0	0	2
7	3	4	5	6	1	1	n/a	1	0	0
8	10	5	12	25	1	4	20	n/a	0	4
9	20	16	19	26	14	18	20	12	n/a	6
10	15	13	14	21	9	13	17	6	0	n/a

Alg. #	Outperformed the others	Title
1	65	Equal
2	42	Linear(CD)
3	68	Linear(C_{R2})
4	146	Linear(C_{HV})
5	26	Log(CD)
6	46	Log(C_{R2})
7	112	Log(C_{HV})
8	19	Metric(CD)
9	0	Metric(C_{R2})
10	20	Metric(C_{HV})

Table 7.18: Summarized table of the Hypervolume results obtained by each MO-CMA-ES variant for the WFG problem up to 8 objectives. The upper part shows the number of times the algorithm outperformed the others. The lower part shows the sum of the times an algorithm outperformed all the others.

Alg.	1	2	3	4	5	6	7	8	9	10
1	n/a	0	1	15	0	2	12	0	2	3
2	2	n/a	4	21	0	2	18	0	2	7
3	1	0	n/a	8	0	0	10	0	3	4
4	0	0	0	n/a	0	0	0	0	0	0
5	13	3	16	22	n/a	9	20	0	5	8
6	2	0	3	14	0	n/a	15	0	1	5
7	1	0	2	2	0	0	n/a	0	0	0
8	10	5	13	24	1	8	22	n/a	4	9
9	16	14	17	21	10	15	20	5	n/a	10
10	6	6	13	15	3	9	16	1	0	n/a

Alg. #	Outperformed the others	Title
1	51	Equal
2	28	Linear(CD)
3	69	Linear(C_{R2})
4	142	Linear(C_{HV})
5	14	Log(CD)
6	45	Log(C_{R2})
7	133	Log(C_{HV})
8	6	Metric(CD)
9	17	Metric(C_{R2})
10	46	Metric(C_{HV})

7.4.1.4 Comparison to classical MOEAs and their CMA-ES variants

From the previous experimental studies, we identified that the best TWFs to be used with MO-CMA-ES are the Linear(C_{HV}) and Metric(C_{HV}). In this section, we compare the MO-CMA-ES using these two TWFs (referred to as just linear and metric from now on) to three classical and well-known MOEAs: NSGA-II (Deb et al., 2000), SPEA2 (Zitzler et al., 2002) and IBEA (Zitzler and Künzli, 2004).

Moreover, we included in the comparison three variants of these MOEAs that were created by applying one single modification in their original frameworks: the replacement of the

standard crossover operator by the CMA-ES using the Log(C_{HV}) TWF. This particular TWF was selected due to the good results of the contributing hypervolume indicator and because of the stable performance of the log weight distribution, which achieves competitive results in most of the scenarios studied.

The parameters used in this experimental study are:

- NSGA-II, IBEA and SPEA2: Binary-tournament
- IBEA: Hypervolume indicator; kappa (k) = 0.05; ρ = 1.1

The common parameters used for all standard MOEAs are: Mutation rate: 1 %, Crossover rate: 100 %, and Distribution index (Deb and Agrawal, 1995) (η_c - mutation and recombination): 20. All the algorithms were allowed to run for 50000 function evaluations. The population size was set to 100. It is worth noting that the archive size used in SPEA2 and MO-CMA-ES variants was set equal to the population size.

Regarding the comparison between distinct recombination operators, we use a baseline setting for the MOEAs, where it is applied the Simulated Binary Crossover (SBX) and Polynomial mutation (Deb and Agrawal, 1995) as crossover and mutation operators, respectively.

In the first part of this experimental study, we use the Comparing Continuous Optimizers (COCO) (Hansen et al., 2016) framework. This framework was developed to automate the task of conducting scientifically sound experimental studies involving numerical optimizers. The platform provides benchmark suites, experimental templates and tools for processing and visualizing the outcome of one or more optimizers. The processing of quality indicators is based on runtimes, measured in number of objective function evaluations to reach one or several quality indicator target values. Recently, the platform was rewritten to deal with multi-objective problems and optimizers, for doing so, a new bi-objective suite of benchmark problems and new performance assessment mechanisms were proposed.

The new bi-objective benchmark suite was called “bbob-biobj". This suite was created by combining a subset of the 24 single-objective problems of the original “bbob" test suite. Combining the 24 original functions without permutations would result in 300 problems. However, having so many functions would be impracticable in terms of the overall running time. Hence, two representative functions of each of the five domains of difficulty available were chosen in order not to introduce bias toward any specific domain. These pairwise combinations resulted in 55 bi-objective functions of the final “bbob-biobj" suite.

The 55 functions are grouped in 15 classes according to the domains of difficulty of its component subproblems as presented in Table 7.19. Each of these functions is provided in six dimensions (2, 3, 5, 10, 20 and 40) and with a large number of possible instances (Brockhoff et al., 2016).

One of the most remarkable characteristics of COCO is its new performance assessment mechanism that considers a quality indicator based on the hypervolume of the external archive A_t instead of the objective value of a single-objective function. This external archive contains all non-dominated solutions obtained so far in an algorithm run.

The target values are based on a target precision ΔI and a reference hypervolume indicator value I^{ref}, which is an approximation of the I_{HV}^{COCO} indicator value of the Pareto front (Brockhoff et al., 2016). The results of the performance assessment are presented as empirical distribution functions, where the proportion of problems solved within a specified budged (in x-axis) is displayed.

The experiments executed with the COCO framework have been exhaustive. We have evaluated the algorithms for all the 55 functions. In addition, to evaluate the scalability in the

Table 7.19: Classes of functions included in the COCO benchmark

Class	Functions	Domain F1	Domain F2
1	f1,f2,f11	separable	separable
2	f3,f4,f12,f13	separable	moderate
3	f5,f6,f14,f15	separable	ill-conditioned
4	f7,f8,f16,f17	separable	multi-modal
5	f9,f10,18,f19	separable	weakly-structured
6	f20,f21,f28	moderate	moderate
7	f22,f23,f29,f30	moderate	ill-conditioned
8	f24,f25,f31,f32	moderate	multi-modal
9	f26,f27,f33,f34	moderate	weakly-structured
10	f35,f36,f41	ill-conditioned	ill-conditioned
11	f37,f38,f42,f43	ill-conditioned	multi-modal
12	f39,f40,f44,f45	ill-conditioned	weakly-structured
13	f46,f47,f50	multi-modal	multi-modal
14	f48,f49,f51,f52	multi-modal	weakly-structured
15	f53,f54,f55	weakly-structured	weakly-structured

number of variables, we have considered all the following dimensions, $DIM \in \{2, 3, 5, 10, 20, 40\}$. The measure used to compare the algorithms is the empirical cumulative distribution (ECD) of the runtime for a predefined number of accuracy targets with respect to the quality indicator (I_{HV}^{COCO}). Basically, this metric measures how many evaluations requires the algorithm to get closer to a Pareto front approximation that approximates the value of the quality indicator to a given accuracy (target). The best value of the metric is 1, meaning that the algorithm was able to reach all the 58 increasingly stricter accuracy targets. The algorithm that reaches these accuracy targets with less function evaluations is considered to be more efficient. Figure 7.2 summarizes the results achieved by all the algorithms. Each sub-figure corresponds to the results for one of the six dimensions considered.

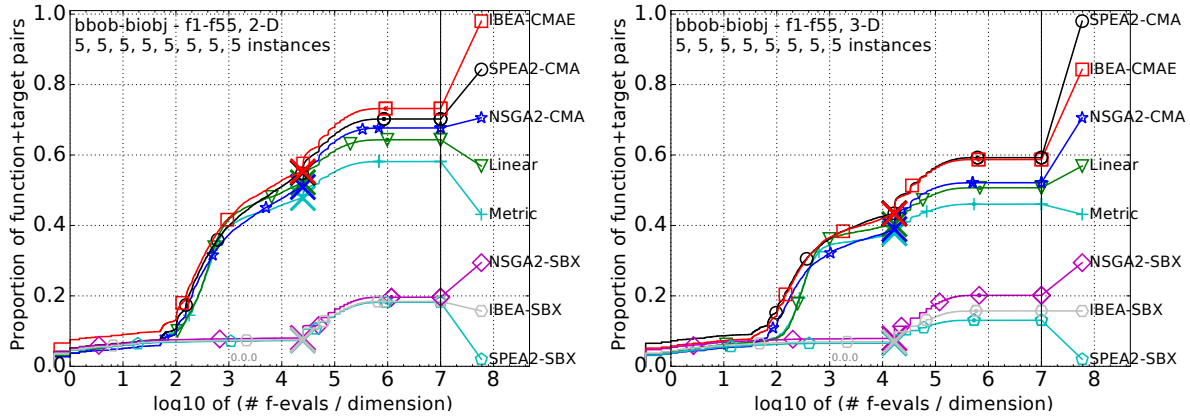
The first pattern that can be appreciated from the analysis of the figure is that SBX performs worse than all algorithms that apply the CMA-ES for all dimensions. The only exception is for 40D, where NSGA2-SBX outperforms its CMA-ES counterpart and the MO-CMA-ES variants. The proportion of accuracy targets that the SBX variants are able to reach is never higher than 0.2. This trend is particularly noticeable for low dimensions ($DIM \leq 10$) for which the difference with the CMA-ES variants is clear. Regarding the CMA-ES behavior, their variants reach above the 0.7 accuracy threshold for $DIM = 2$ and are always above 0.2 for all dimensions.

As expected, when the number of dimensions is increased the proportion of targets reached by all the algorithms is lower. A third fact revealed by the experiments is that while IBEA and SPEA are consistently the best selection strategies for CMAEs, the results for SBX are different. For the traditional GA operator, NSGA2, in general, produces better results than the other selection methods. The influence of the selection operator is so important that for 40 dimensions the NSGA2-SBX outperforms NSGA2-CMA, as well as our MO-CMA-ES variants.

Regarding only the MO-CMA-ES variants, a pattern that arises is that linear always outperforms the metric up to five dimensions, however when the number of dimensions is over five, the metric variant consistently outperforms the other. Another finding is that although the MO-CMA-ES loses for the SPEA2, IBEA and NSGA2 variants when using CMA-ES, they outperform their SBX counterparts in all cases, except for NSGA2-SBX for 40 dimensions.

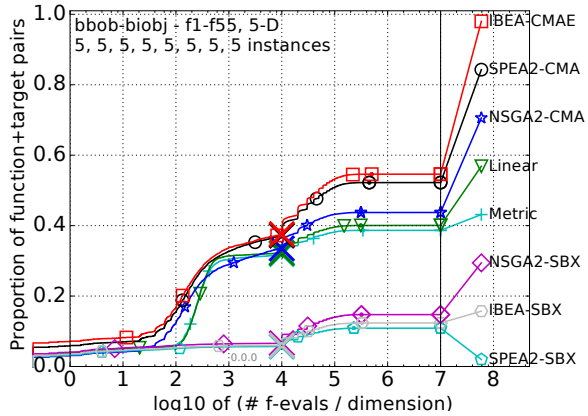
A second round of experiments is now conducted involving the same algorithms, but on the DTLZ family of benchmark problems. Here we removed the linear MO-CMA-ES variant, since in the previous section the metric TWF outperformed it for this family of problems.

The results of these experiments measured by the IGD_p and hypervolume are presented in Tables 7.20 and 7.22 respectively, while the summarized results of IGD_p and hypervolume are presented in Tables 7.21 and 7.23.

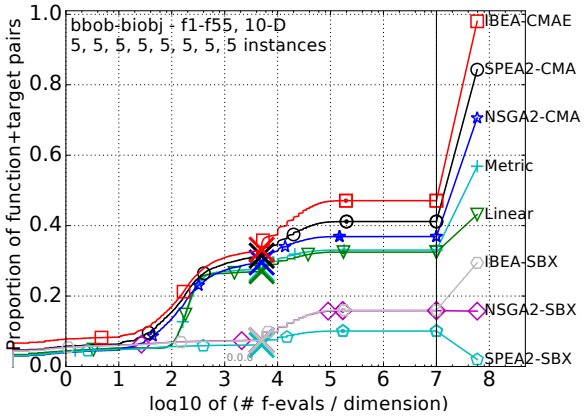


(a) 02D

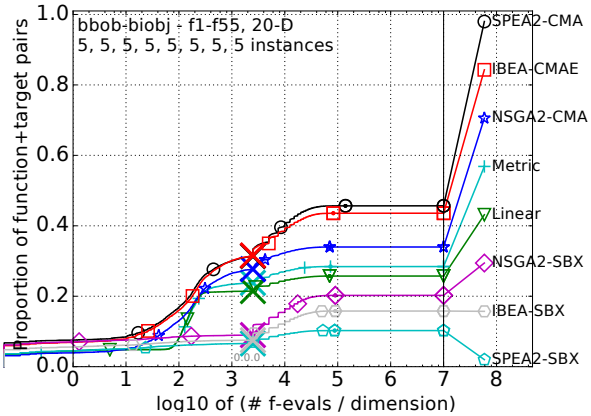
(b) 03D



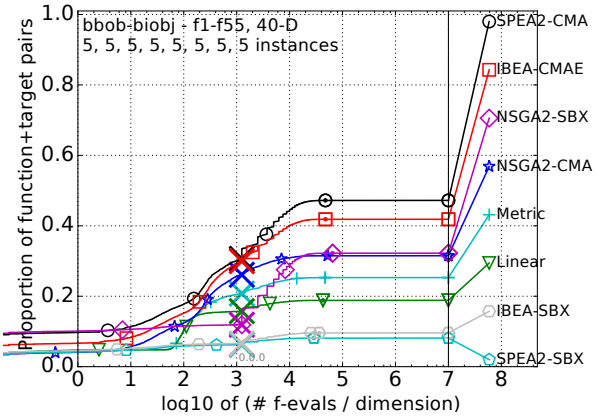
(c) 05D



(d) 10D



(e) 20D



(f) 40D

Figure 7.2: Empirical cumulative distribution (ECD) of simulated (bootstrapped) runtimes, measured in number of objective function evaluations, divided by decision space dimension (FEvals/DIM) for the 58 targets of all algorithms $\{-10^{-4}, -10^{-4.2}, -10^{-4.4}, -10-^{4.6}, -10^{-4.8}, -10^{-5}, 0, 10^{-5}, 10^{-4.9}, 10^{-4.8}, \ldots, 10^{-0.1}, 10^{0}\}$. Results for all 55 functions and all dimensions.

Regarding the IGD_p indicator, we can highlight that for two objectives, in general, the SBX variants achieved the best performances, losing to CMA-ES variants only on problem

DTLZ6 and being tied with our MO-CMA-ES on DTLZ5. For three objectives, the CMA-ES variants become more competitive. In this scenario, the SBX variants only outperformed all CMA-ES variants on problems DTLZ1 and DTLZ3. MO-CMA-ES metric performed the best on problem DTLZ5.

Table 7.20: Mean ranks of the IGD_p obtained for the DTLZ problems by each of the classical MOEA variants and MO-CMA-ES as used in the Kruskal-Wallis test. Final ranks, presented in parentheses assigned according to mean ranks.

Obj.	Algorithms	DTLZ1	DTLZ2	DTLZ3	DTLZ4	DTLZ5	DTLZ6	DTLZ7
2	Metric	138.47 (5.50)	67.83 (3.00)	181.37 (5.50)	127.60 (4.50)	**57.90 (2.00)**	86.07 (4.00)	136.13 (5.50)
	IBEA-CMAES	149.47 (5.50)	172.37 (6.50)	137.13 (5.50)	104.27 (4.00)	189.97 (6.50)	122.23 (4.50)	173.43 (6.00)
	NSGA2-CMAES	157.40 (5.50)	110.73 (4.00)	143.00 (5.50)	76.00 (3.00)	122.27 (5.00)	**39.70 (1.50)**	83.63 (3.00)
	SPEA2-CMAES	155.73 (5.50)	119.00 (4.50)	140.50 (5.50)	84.27 (3.50)	109.40 (4.00)	**35.10 (1.50)**	50.60 (2.00)
	IBEA-SBX	**55.47 (2.00)**	187.63 (6.50)	**34.63 (2.00)**	114.00 (4.00)	166.53 (6.00)	125.77 (4.50)	154.57 (5.50)
	NSGA2-SBX	**50.47 (2.00)**	61.53 (2.00)	**51.57 (2.00)**	**51.10 (2.00)**	66.80 (2.50)	189.37 (7.00)	108.60 (4.50)
	SPEA2-SBX	**31.50 (2.00)**	**19.40 (1.50)**	**50.30 (2.00)**	181.27 (7.00)	**25.63 (2.00)**	140.27 (5.00)	**31.53 (1.50)**
3	Metric	139.70 (5.50)	104.57 (4.00)	174.03 (5.50)	168.33 (6.00)	**19.03 (1.50)**	85.30 (3.50)	**68.90 (2.50)**
	IBEA-CMAES	151.63 (5.50)	52.27 (2.00)	136.07 (5.50)	111.53 (4.00)	47.27 (2.00)	83.83 (3.50)	**81.47 (2.50)**
	NSGA2-CMAES	158.73 (5.50)	69.67 (3.00)	148.23 (5.50)	69.80 (2.50)	70.33 (3.00)	119.20 (4.50)	**54.67 (2.50)**
	SPEA2-CMAES	151.90 (5.50)	**15.50 (1.50)**	143.57 (5.50)	**50.87 (2.00)**	105.37 (4.00)	**55.27 (2.00)**	**39.73 (2.50)**
	IBEA-SBX	**42.57 (2.00)**	179.13 (6.00)	**23.20 (2.00)**	130.87 (5.00)	135.50 (5.00)	104.53 (4.00)	136.50 (5.50)
	NSGA2-SBX	**61.10 (2.00)**	181.27 (6.00)	**66.10 (2.00)**	**29.90 (2.00)**	195.07 (6.50)	163.37 (6.00)	162.13 (6.00)
	SPEA2-SBX	**32.87 (2.00)**	136.10 (5.50)	**47.30 (2.00)**	177.20 (6.50)	165.93 (6.00)	127.00 (4.50)	195.10 (6.50)

When considering the summarized IGD_p results, all algorithms appear statistically tied, however if we consider only the mean ranks as tiebreakers, we can highlight IBEA-SBX and SPEA2-SBX as those with the best mean ranks. Following the same criterion, we can highlight NSGA2-SBX as worst performing variant.

The hypervolume results are similar to the IGD_p, where for two objectives the SBX variants are outperformed only on problem DTLZ6 by NSGA2-CMAES and SPEA2-CMAES. When considering three objectives, the CMA-ES variants become more competitive, achieving the best results on problems DTLZ2, DTLZ5, DTLZ6 and DTLZ7. Moreover, the IBEA-CMAES appears statistically tied with IBEA-SBX in DTLZ4.

The summarized hypervolume results show all algorithms tied, as the IGD_p. If we use the mean ranks as tiebreakers, IBEA-SBX appears as the best algorithm, while the MOPSO-CMA-ES metric appears as the worst.

Table 7.21: Overall ranks of IGD_p as used in the Friedman test for all the DTLZ problems and numbers of objectives. Final ranks, presented in parentheses assigned according to the ranks.

Metric	IBEA-CMAES	NSGA2-CMAES	SPEA2-CMAES	IBEA-SBX	NSGA2-SBX	SPEA2-SBX
59.0 (4.0)	**59.0 (4.0)**	**57.0 (4.0)**	**54.0 (4.0)**	**51.0 (4.0)**	**61.0 (4.0)**	**51.0 (4.0)**

A fact that needs to be highlighted is that all the CMA-ES variants achieve very poor results in the problems DTLZ1 and DTLZ3. These problems have many local optimal fronts, which challenge the ability of the algorithms to converge to the true Pareto front by avoiding premature convergence. In Figure 7.3, we plot the best 2D front (according to hypervolume) found by each CMA-ES variant studied here. This result indicates that all algorithms failed to converge to the true Pareto front in these problems, especially on DTLZ3, however further study needs to be conducted to exactly investigate the causes behind this behavior.

7.4.1.5 Summary of the results

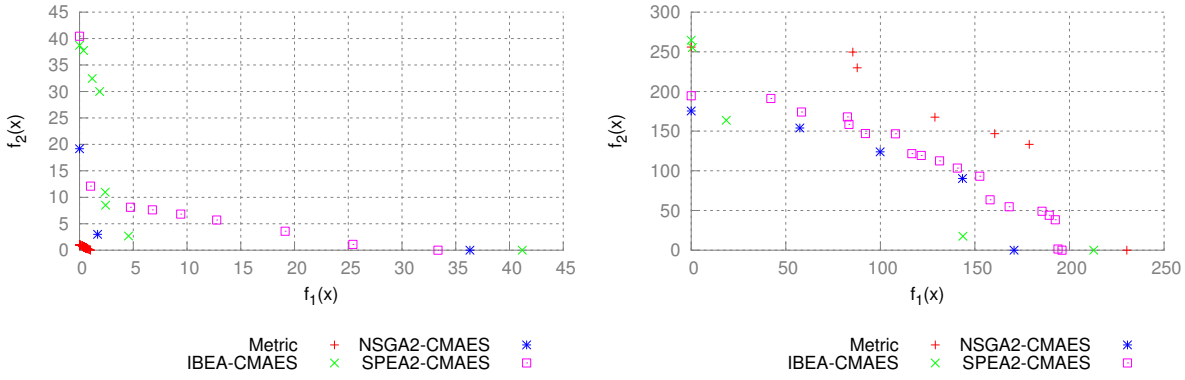
We can summarize and discuss the results achieved by each TWF in three parts, based on their characteristics: The equal, which is unique since it does not use a ranking strategy; the

Table 7.22: Mean ranks of the hypervolume obtained for the DTLZ problems by each of the classical MOEA variants and MO-CMA-ES as used in the Kruskal-Wallis test. Final ranks, presented in parentheses assigned according to mean ranks.

Obj.	Algorithms	DTLZ1	DTLZ2	DTLZ3	DTLZ4	DTLZ5	DTLZ6	DTLZ7
2	Metric	171.57 (5.50)	86.50 (3.00)	194.00 (7.00)	127.67 (5.00)	86.40 (3.00)	78.87 (2.50)	162.97 (6.00)
	IBEA-CMAES	137.80 (5.50)	146.37 (6.00)	140.60 (5.00)	84.87 (4.00)	144.60 (6.00)	107.60 (4.50)	114.90 (4.50)
	NSGA2-CMAES	146.13 (5.50)	170.50 (6.00)	127.83 (5.00)	121.23 (4.50)	173.10 (6.00)	**41.30 (2.00)**	118.27 (5.00)
	SPEA2-CMAES	146.03 (5.50)	173.20 (6.00)	139.57 (5.00)	130.53 (5.00)	172.47 (6.00)	**33.97 (2.00)**	169.50 (6.50)
	IBEA-SBX	**60.00 (2.00)**	56.03 (2.50)	**35.13 (2.00)**	**31.57 (1.50)**	56.03 (2.50)	137.37 (5.00)	**58.70 (2.00)**
	NSGA2-SBX	**46.57 (2.00)**	87.10 (3.00)	**51.13 (2.00)**	71.13 (2.00)	87.10 (3.00)	191.33 (6.50)	**49.87 (2.00)**
	SPEA2-SBX	**30.40 (2.00)**	**18.80 (1.50)**	**50.23 (2.00)**	171.50 (6.00)	**18.80 (1.50)**	148.07 (5.50)	**64.30 (2.00)**
3	Metric	188.30 (6.50)	105.40 (4.00)	188.77 (6.50)	155.10 (6.50)	**18.87 (1.50)**	86.97 (3.50)	**86.93 (2.50)**
	IBEA-CMAES	135.73 (5.00)	**15.50 (1.50)**	127.47 (5.00)	**51.97 (1.50)**	43.50 (2.00)	92.03 (3.50)	**46.30 (2.50)**
	NSGA2-CMAES	144.13 (5.50)	75.60 (3.00)	153.27 (5.50)	105.80 (4.00)	74.20 (3.00)	119.63 (4.50)	**65.23 (2.50)**
	SPEA2-CMAES	133.30 (5.00)	45.50 (2.00)	131.77 (5.00)	105.40 (4.00)	105.43 (4.00)	**55.47 (2.00)**	**46.37 (2.50)**
	IBEA-SBX	**52.10 (2.00)**	194.67 (6.50)	**23.80 (2.00)**	**20.13 (1.50)**	171.63 (6.00)	102.00 (4.00)	162.77 (6.00)
	NSGA2-SBX	**55.87 (2.00)**	166.33 (6.00)	**66.40 (2.00)**	105.03 (4.00)	189.37 (6.50)	159.40 (6.00)	147.43 (6.00)
	SPEA2-SBX	**29.07 (2.00)**	135.50 (5.00)	**47.03 (2.00)**	195.07 (6.50)	135.50 (5.00)	123.00 (4.50)	183.47 (6.00)

Table 7.23: Overall ranks of hypervolume as used in the Friedman test for all the DTLZ problems and numbers of objectives. Final ranks, presented in parentheses assigned according to the ranks.

Metric	IBEA-CMAES	NSGA2-CMAES	SPEA2-CMAES	IBEA-SBX	NSGA2-SBX	SPEA2-SBX
69.0 (4.0)	**52.0 (4.0)**	**61.0 (4.0)**	**62.0 (4.0)**	**38.0 (4.0)**	**62.0 (4.0)**	**48.0 (4.0)**



(a) DTLZ1 (b) DTLZ3

Figure 7.3: Best front (according to HV indicator) obtained by each CMA-ES variant on problems DTLZ1 and DTLZ3.

functions using the three distinct metrics, and the functions using the three remaining weighting schemes.

Starting by the equal, if we consider the summarized tables 7.7, 7.8, 7.9 and 7.10 equal only outperformed the other TWFs more times in Table 7.9, where C_{HV} was not used as ranking method. More specifically through Tables 7.3 and 7.4, we can see that most of the times equal appeared among the best algorithms was up to eight objectives (17 times). When we include the TWFs ranked by the C_{HV} metric, this number drops to 4 times, from which, despite not having statistical differences from the best, it has the absolute smallest rank in only one case (WFG1 $m = 8$). This analysis indicates that using an equal weight distribution can be advantageous only when we do not have reliable metrics to rank the solutions.

Among the functions using the three different ranking indicators, the results in all tested cases show that C_{HV} performs the best, since within all blocks of weighting strategies C_{HV} statistically outperforms the other algorithms more times. The relevance of these results is

twofold: first it confirms that if we use a powerful quality indicator to rank the solutions, the model learns these preferences introduced by the indicator and generates solutions following this criterion; secondly, it was expected (although not guaranteed) that if the model learns through the preferences of hypervolume, it would generate good solutions according to this metric. However we compared the final approximations using the IGD_p indicator and C_{HV} still obtained the best results, which means that the model is really generating better solutions, instead of simply exploiting information about the metric to sample solutions which are biased toward the preferred regions of the hypervolume.

However, using the exact hypervolume calculation for problems with more than eight objectives is computationally very expensive. Approximate versions do exist, but generating enough samples in order to achieve reliable results in each iteration is computationally expensive as well, this is why we used C_{HV} only up to eight objectives and used the approximate hypervolume only to evaluate the final front generated by each variant of the algorithm. When considering the other metrics used (C_{R2} and CD), we can observe that in most of the cases it is preferable to use C_{R2} over CD. This result can be explained by the fact that R2 is a metric that measures both convergence and diversity, while CD is only an estimator of diversity. This difference can be seen in most of the tables, where CD is among the best algorithms more times for three objectives (where the pressure toward the Pareto front is stronger) than for other numbers of objectives.

A problem often seen in C_{R2} is that several solutions present a value of zero. This happens when more than one solution is close to a single weight vector, which makes the closer solution contribute to the general R2, but the farthest has a contribution of zero. A possible way to ease this problem would be to have more weight vectors, but this can change the relation convergence versus diversity of the algorithm. This would happen because if we have more weight vectors, more solutions in a same region will contribute to the R2 and consequently to the model update. This would generate even more solutions in that region, but a detailed analysis on the effect of the density of weight vectors for the C_{R2} indicator is material for future works.

The third characteristic to be observed on the TWFs is the strategy used to assign weights to the solutions ranked (equal does not use the ranks), where we have the three variants: linear, log and metric. Among these three variants, we can state that linear introduces less selection pressure toward the best ranked solutions, since the differences in weights are smaller. The same way, log is an intermediate, and in general, metric applies more selection pressure since the differences among the first and last solutions usually are larger (see Figure 7.1). This trend is reflected in the results obtained, where for the DTLZ problems the number of times each of these schemes outperforms the others (including themselves) can be expressed as linear < log < metric in all the summarized tables except in hypervolume when considering up to 8 objectives. An inverse trend can be observed when considering the WFG problems, where these numbers can be expressed as linear > log > metric in all the summarized tables except in hypervolume when considering up to 20 objectives. This inverse impact observed about the selection pressure applied by both functions is an important fact, however so far we have not been able to identify a particular reason for this behavior. Additional research will be conducted to investigate this difference and gain further insight into the effects of using TWFs.

The discussion of the results obtained by the comparison of the MO-CMA-ES using the best TWFs to the classical MOEA algorithms and their variants has to be divided in two parts: the first one is the comparison using the COCO framework, where although the MO-CMA-ES variants were not able to outperform the MOEA variants that use CMA-ES, they were able to outperform the original MOEAs using SBX in all cases, except NSGA2 for 40D. These results indicate that the MO-CMA-ES variants proposed are promising algorithms for complicated problems, however other possibilities have to be investigated to improve their performance.

The second part of this discussion is concentrated on the results obtained using the DTLZ benchmark problems. In this case, for two objectives, the CMA-ES versions were outperformed in most cases, however when the number of objectives was increased to three, the performance of the CMA-ES variants increased and they became competitive. Among the CMA-ES variants, the IBEA-CMAES and SPEA2-CMAES achieved the best performances, while our MO-CMA-ES achieved poor results in this comparison. Another fact that needs to be highlighted is that all the CMA-ES variants achieve very poor results in the problems DTLZ1 and DTLZ3. This behavior can be explained by the fact that these problems have many local optimal fronts and the CMA-ES variants have a tendency of being trapped by these local optima on these problems.

7.4.1.6 Summary of the results

The injection of information from good solutions is a fundamental step for the CMA-ES algorithm, since this information will be used to create and evolve a model of the problem being explored. In Pareto based algorithms, several methods can be used to determine how "good" a non-dominated solution is and, thus, can be used to rank the solutions of a Pareto set. However, each method can value different characteristics of the solutions, leading to different models being learned. Moreover, the pressure introduced by a ranking method on the model can be adjusted by appropriately selecting a weighting function over the ranked solutions.

As far as we know, this is the first work to present the idea of using Transfer Weight Functions (TWFs) as flexible components to manage the incorporation of information into a Pareto based MO-CMA-ES, moreover, we compared the performance of different TWFs. The TWF components used here included three different methods to rank the solutions, each one based on a well-known quality indicator. Furthermore, four different ways of distributing weights to the solutions were used, one of the options was giving them equal weights, so there is no influence from the ranking method, the other three alternatives give higher weights to better ranked solutions. Each combination of weighting strategy and ranking metric composed as TWF, except the equal weight distribution which was treated separately, hence, we end up with 10 different TWFs.

In order to assess the influence of the proposed TWFs on the MO-CMA-ES algorithm, we conducted an experimental study comparing the TWFs on two sets of popular benchmark problems on a total of 16 functions. Additionally, we used different numbers of objectives, ranging from 3 to 20, to observe the scalability of these TWFs in the many-objective scenario. The results obtained were evaluated using two different quality indicators and the values measured by them were submitted to a statistical test to look for statistically significant differences.

Our results indicate that, in general, the equal TWF does not produce the best results, therefore, it is useful to apply a ranking method and unequal weights for the solutions according to its quality. When comparing the remaining strategies to weight the ranked solutions, we saw that a good choice can be problem-dependent, since opposed trends were found when analyzing both families of problems: for WFG, in most cases, it is best to use the linear approach, while for DTLZ the metric approach performed better.

When considering the three different ranking indicators, we clearly obtained the best results by using C_{HV}, however, due to the high computational cost to calculate it exactly or approximate it reliably when using many-objectives, we limited its use up to eight objectives. In the scenarios where the C_{HV} was absent, in most cases C_{R2} outperformed CD, this difference was smaller when we considered three objectives, but as the number of objectives increased and, consequently, the influence of the Pareto dominance weakened, the differences in favor of the

C_{R2} indicator increased, since it takes into consideration both the convergence and the diversity, while CD is only a diversity estimator.

In a comparison with classical MOEAs and their CMA-ES variants, our MO-CMA-ES variants were able to outperform the classical MOEAs in most of the cases in the COCO framework, which presents very hard combinations of single-objective optimization problems. When using the DTLZ family of problems, the MO-CMA-ES metric was outperformed in most of the investigated cases.

7.4.2 Empirical studies involving the multiple model MO-CMA-ES

The main goal of this experimental study is to evaluate the performance of the MO-CMA-ES-rankOne and MO-CMA-ES-rankMu algorithm variants, and to compare their performance to the best performing algorithm from the previous experimental study: IBEA-SBX.

The common parameters used for the IBEA-SBX are: selection by binary-tournament, hypervolume indicator, kappa (k) = 0.05; ρ = 1.1. Mutation rate: 1 %, Crossover rate: 100 %, and Distribution index (Deb and Agrawal, 1995) (η_c - mutation and recombination): 20.

IBEA-SBX and the bounded population variants of MO-CMA-ES used a population size of 100. The rank mu variants used a neighborhood size of $T = 20$ for all problems and objective numbers. All the algorithms were allowed to run for 50000 function evaluations.

7.4.2.1 Comparison including the unbounded variants of MO-CMA-ES

We now present the results of the experiments conducted using the DTLZ family of benchmark problems for 2 and 3 objectives. For these experiments, we used two variants of MO-CMA-ES-rankOne as well as two variants of MO-CMA-ES-rankMu. One of the variants has an unbounded population (identified as "UB"), meaning that all the non-dominated solutions are kept without restriction, the other version uses the crowding distance archiver (identified as "CD") to keep only 100 individuals in the population.

The results obtained by the algorithms, measured by the IGD_p are presented in Table 7.24, and the hypervolume results are presented in Table 7.26. Since it can be hard to visualize differences among the results in the general tables, summarized results are presented in Tables 7.25 and 7.27 for the IGD_p and hypervolume results respectively.

Table 7.24: Mean ranks of the IGD_p as used in the Kruskal-Wallis test for the DTLZ problems. Final ranks, presented in parentheses assigned according to mean ranks.

Obj.	Algorithms	DTLZ1	DTLZ2	DTLZ3	DTLZ4	DTLZ5	DTLZ6	DTLZ7
2	IBEA-SBX	**16.40 (1.00)**	135.50 (4.50)	**22.37 (1.00)**	119.50 (5.00)	135.50 (4.50)	**64.37 (2.00)**	109.90 (4.00)
	MO-CMA-ES$_{CD}$-rankMu	110.18 (4.50)	71.17 (3.00)	85.02 (3.50)	81.70 (3.00)	75.50 (3.00)	**43.87 (2.00)**	**17.07 (1.00)**
	MO-CMA-ES$_{UB}$-rankMu	110.22 (4.50)	**33.83 (1.50)**	85.05 (3.50)	62.57 (2.50)	**15.50 (1.50)**	**38.67 (2.00)**	48.67 (2.00)
	MO-CMA-ES$_{CD}$-rankOne	66.37 (2.50)	105.20 (4.50)	93.93 (3.50)	80.73 (3.00)	105.50 (4.00)	115.63 (4.50)	93.67 (4.00)
	MO-CMA-ES$_{UB}$-rankOne	74.33 (2.50)	**31.80 (1.50)**	91.13 (3.50)	**33.00 (1.50)**	45.50 (2.00)	114.97 (4.50)	108.20 (4.00)
3	IBEA-SBX	**15.50 (1.00)**	135.50 (4.50)	**15.50 (1.00)**	45.13 (2.00)	135.50 (4.50)	86.50 (4.00)	135.50 (5.00)
	MO-CMA-ES$_{CD}$-rankMu	89.45 (3.50)	75.50 (3.50)	111.95 (4.50)	114.83 (4.50)	45.50 (2.00)	**40.53 (1.50)**	**42.67 (2.00)**
	MO-CMA-ES$_{UB}$-rankMu	90.52 (3.50)	**23.50 (1.50)**	108.45 (4.50)	118.33 (4.50)	**15.50 (1.50)**	**20.47 (1.50)**	**53.43 (2.00)**
	MO-CMA-ES$_{CD}$-rankOne	88.77 (3.50)	105.50 (4.00)	69.57 (2.50)	70.43 (2.50)	105.50 (4.00)	114.07 (4.00)	95.23 (4.00)
	MO-CMA-ES$_{UB}$-rankOne	93.27 (3.50)	**37.50 (1.50)**	72.03 (2.50)	**28.77 (1.50)**	75.50 (3.00)	115.93 (4.00)	**50.67 (2.00)**

Considering the IGD_p results, we can identify that the convergence issues of the CMA-ES variants seen in the previous sections on problems DTLZ1 and DTLZ3 remain, since they are outperformed in these problems in all the cases. Besides that, the CMA-ES variants present the best performance in all the problems with two and three objectives, although they statistically tie with IBEA on problem DTLZ6.

Table 7.25: Overall ranks of IGD_p as used in the Friedman test for all the DTLZ problems and numbers of objectives. Final ranks, presented in parentheses assigned according to the ranks.

IBEA-SBX	MO-CMA-ES$_{CD}$-rankMu	MO-CMA-ES$_{UB}$-rankMu	MO-CMA-ES$_{CD}$-rankOne	MO-CMA-ES$_{UB}$-rankOne
41.0 (3.0)	**43.0 (3.0)**	**39.0 (3.0)**	**47.0 (3.0)**	**40.0 (3.0)**

In the summarized table, all algorithms appear tied, this is mostly because of the influence of the problems DTLZ1 and DTLZ3 that benefits the IBEA. Considering the general ranks as tiebreakers, the unbounded versions of MO-CMA-ES-rankMu and MO-CMA-ES-rankOne achieved the best results respectively.

Table 7.26: Mean ranks of the hypervolume as used in the Kruskal-Wallis test for the DTLZ problems. Final ranks, presented in parentheses assigned according to mean ranks.

Obj.	Algorithms	DTLZ1	DTLZ2	DTLZ3	DTLZ4	DTLZ5	DTLZ6	DTLZ7
2	IBEA-SBX	**15.67 (1.00)**	105.50 (4.50)	**15.50 (1.00)**	80.50 (3.00)	105.50 (4.50)	**67.17 (2.00)**	**18.07 (1.50)**
	MO-CMA-ES$_{CD}$-rankMu	119.15 (4.50)	72.93 (2.50)	104.55 (3.50)	87.57 (3.50)	72.93 (2.50)	**45.17 (2.00)**	**44.10 (1.50)**
	MO-CMA-ES$_{UB}$-rankMu	119.62 (4.50)	**15.50 (1.00)**	104.58 (3.50)	62.23 (3.00)	**15.50 (1.00)**	**37.23 (2.00)**	74.77 (3.00)
	MO-CMA-ES$_{CD}$-rankOne	58.63 (2.50)	135.50 (4.50)	76.43 (3.50)	115.70 (4.50)	135.50 (4.50)	114.08 (4.50)	115.17 (4.50)
	MO-CMA-ES$_{UB}$-rankOne	64.43 (2.50)	48.07 (2.50)	76.43 (3.50)	**31.50 (1.00)**	48.07 (2.50)	113.85 (4.50)	125.40 (4.50)
3	IBEA-SBX	**15.53 (1.00)**	135.50 (4.50)	**15.50 (1.00)**	**27.40 (1.50)**	135.50 (4.50)	79.63 (3.00)	135.50 (5.00)
	MO-CMA-ES$_{CD}$-rankMu	105.75 (4.50)	75.50 (3.00)	118.53 (4.50)	111.07 (4.50)	45.50 (2.00)	**44.50 (1.50)**	**24.43 (1.50)**
	MO-CMA-ES$_{UB}$-rankMu	106.78 (4.50)	**15.50 (1.50)**	116.60 (4.50)	120.90 (4.50)	**15.50 (1.50)**	**16.50 (1.50)**	49.43 (2.00)
	MO-CMA-ES$_{CD}$-rankOne	74.93 (2.50)	105.50 (4.00)	61.90 (2.50)	78.27 (3.00)	105.50 (4.00)	117.42 (4.50)	101.13 (4.00)
	MO-CMA-ES$_{UB}$-rankOne	74.50 (2.50)	45.50 (2.00)	64.97 (2.50)	**39.87 (1.50)**	75.50 (3.00)	119.45 (4.50)	67.00 (2.50)

Table 7.27: Overall ranks of hypervolume as used in the Friedman test for all the DTLZ problems and numbers of objectives. Final ranks, presented in parentheses assigned according to the ranks.

IBEA-SBX	MO-CMA-ES$_{CD}$-rankMu	MO-CMA-ES$_{UB}$-rankMu	MO-CMA-ES$_{CD}$-rankOne	MO-CMA-ES$_{UB}$-rankOne
36.0 (3.0)	**45.0 (3.0)**	**40.0 (3.0)**	**51.0 (3.0)**	**38.0 (3.0)**

In the results measured by hypervolume, the IBEA appears more competitive, where besides problems DTLZ1 and DTLZ3, it ties with the CMA-ES variants on problems DTLZ6 and DTLZ7 for two objectives and on problem DTLZ4 for three. In the summarized table, again no statistical differences were found, however, based on the general ranks IBEA performed the best, followed closely by MO-CMA-ES$_{UB}$-rankOne. However, as in the IGD_p case, this result is mostly influenced by the DTLZ1 and DTLZ3 problems. Regarding the MO-CMA-ES variants, in general, the unbounded variants outperforms the ones that employs the archiver.

7.4.2.2 Comparison without the unbounded variants of MO-CMA-ES

In the previous section, we compared four variants of MO-CMA-ES with IBEA, and, in general, the unbounded variants achieved the best results. Although in many situations there is no problem in using an unbounded algorithm, there are some cases where using them is harder due to hardware limitations. These situations include:

- *Increased number of objectives*, where the number of non-dominated solutions can increase rapidly.
- *Large number of iterations*, where, in general, the number of non-dominated solutions increases along the iterations.
- *Increased number of decision variables*, where the memory requirements to store the CMA-ES models of a large population can become prohibitive.

Due to these reasons, one might want to avoid unbounded algorithms, even at the sacrifice of searching quality. Hence, in this section, we present a comparison like the previous, but without the inclusion of the unbounded versions of the MO-CMA-ES variants. Since we are conducting comparative analyses of the algorithms based on ranks, the overall patterns can change dramatically when adding or removing algorithms from the pool.

The results of this section are presented in Tables 7.28 and 7.30 featuring the ranks of the IGD_p and hypervolume values. A summarized view of these results are presented in Tables 7.29 and 7.31 for the IGD_p and hypervolume respectively.

Regarding the IGD_p results, we can observe that without considering the unbounded variants, the relative performance of the rank mu variant in general is better than the other two algorithms, both in the two objective and in the three objective problems. Moreover, the IBEA only outperformed the CMA-ES variants on problems DTLZ1 and DTLZ3, and on problem DTLZ4 for three objectives. The summarized table shows all three algorithms statistically tied. The IBEA and the MO-CMA-ES$_{CD}$-rankMu appear tied as best algorithms in the overall ranks as well.

Table 7.28: Mean ranks of the IGD_p as used in the Kruskal-Wallis test for the DTLZ problems. Final ranks, presented in parentheses assigned according to mean ranks.

Obj.	Algorithms	DTLZ1	DTLZ2	DTLZ3	DTLZ4	DTLZ5	DTLZ6	DTLZ7
2	IBEA-SBX	**16.13 (1.00)**	75.50 (3.00)	**18.93 (1.00)**	67.50 (3.00)	75.50 (3.00)	39.97 (2.00)	64.93 (2.50)
	MO-CMA-ES$_{CD}$-rankMu	71.40 (3.00)	**15.50 (1.00)**	55.67 (2.50)	**35.23 (1.50)**	**15.50 (1.00)**	**23.87 (1.00)**	**15.50 (1.00)**
	MO-CMA-ES$_{CD}$-rankOne	48.97 (2.00)	45.50 (2.00)	61.90 (2.50)	**33.77 (1.50)**	45.50 (2.00)	72.67 (3.00)	56.07 (2.50)
3	IBEA-SBX	**15.50 (1.00)**	75.50 (3.00)	**15.50 (1.00)**	**19.53 (1.00)**	75.50 (3.00)	51.50 (2.00)	75.50 (3.00)
	MO-CMA-ES$_{CD}$-rankMu	60.83 (2.50)	**15.50 (1.00)**	71.13 (3.00)	73.30 (3.00)	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**
	MO-CMA-ES$_{CD}$-rankOne	60.17 (2.50)	45.50 (2.00)	49.87 (2.00)	43.67 (2.00)	45.50 (2.00)	69.50 (3.00)	45.50 (2.00)

Table 7.29: Overall ranks of IGD_p as used in the Friedman test for all the DTLZ problems and numbers of objectives. Final ranks, presented in parentheses assigned according to the ranks.

IBEA-SBX	MO-CMA-ES$_{CD}$-rankMu	MO-CMA-ES$_{CD}$-rankOne
27.0 (2.0)	**27.0 (2.0)**	**30.0 (2.0)**

The hypervolume results indicate that the IBEA is more competitive, where besides outperforming the CMA-ES variants on problems DTLZ1 and DTLZ3, it also performs well in the problem DTLZ4 and in DTLZ7 for two objectives. In most of the problems, however, MO-CMA-ES$_{CD}$-rankMu achieves the best results. In this comparison, the rank one variant could not achieve the best results in any problem.

In the summarized table, the three algorithms present no statistically significant differences, however IBEA has the smallest overall rank, followed by the rank mu and the rank one variants respectively. This good result of the IBEA is most certainly motivated by its good results on problems DTLZ1, DTLZ3 and DTLZ4.

Table 7.30: Mean ranks of the hypervolume as used in the Kruskal-Wallis test for the DTLZ problems. Final ranks, presented in parentheses assigned according to mean ranks.

Obj.	Algorithms	DTLZ1	DTLZ2	DTLZ3	DTLZ4	DTLZ5	DTLZ6	DTLZ7
2	IBEA-SBX	**15.63 (1.00)**	45.50 (2.00)	**15.50 (1.00)**	**32.50 (1.50)**	45.50 (2.00)	41.33 (2.00)	**18.30 (1.00)**
	MO-CMA-ES$_{CD}$-rankMu	75.20 (3.00)	**15.50 (1.00)**	67.57 (2.50)	**38.00 (1.50)**	**15.50 (1.00)**	**23.83 (1.00)**	42.70 (2.00)
	MO-CMA-ES$_{CD}$-rankOne	45.67 (2.00)	75.50 (3.00)	53.43 (2.50)	66.00 (3.00)	75.50 (3.00)	71.33 (3.00)	75.50 (3.00)
3	IBEA-SBX	**15.53 (1.00)**	75.50 (3.00)	**15.50 (1.00)**	**17.50 (1.00)**	75.50 (3.00)	48.27 (2.00)	75.50 (3.00)
	MO-CMA-ES$_{CD}$-rankMu	68.27 (2.50)	**15.50 (1.00)**	74.53 (3.00)	70.57 (3.00)	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**
	MO-CMA-ES$_{CD}$-rankOne	52.70 (2.50)	45.50 (2.00)	46.47 (2.00)	48.43 (2.00)	45.50 (2.00)	72.73 (3.00)	45.50 (2.00)

Table 7.31: Overall ranks of hypervolume as used in the Friedman test for all the DTLZ problems and numbers of objectives. Final ranks, presented in parentheses assigned according to the ranks.

IBEA-SBX	MO-CMA-ES$_{CD}$-rankMu	MO-CMA-ES$_{CD}$-rankOne
23.0 (2.0)	**28.0 (2.0)**	**33.0 (2.0)**

7.4.2.3 Summary of the results

The results of this section can be summarized in three parts: the specific case of the DTLZ1 and DTLZ3 problems; the results including the unbounded MO-CMA-ES variants; the results without the unbounded variants.

In all experiments presented, the algorithms that employ the CMA-ES perform poorly in the problems DTLZ1 and DTLZ3, the MO-CMA-ES variants proposed here are no exception. As discussed previously this behavior is due to the premature convergence in such type of problem. Knowing this fact we will not consider these problems in this summary of results.

In the study including the unbounded algorithms, the two unbounded versions presented competitive performance among themselves. In a comparison to the bounded versions, the rank one variant never achieved the best results, while the rank mu variant only had good results in the DTLZ6 and DTLZ7 problems. These problems have a smaller objective space, since DTLZ6 is degenerate and DTLZ7 is discontinuous, hence the smaller number of solutions did not greatly affect the bounded version.

In the study including only the bounded versions of the MO-CMA-ES variants, IBEA only had good results in the problem DTLZ4 (excluding DTLZ1 and DTLZ3). In the other four problems, the rank mu variant outperformed all the other algorithms with a very large difference in the general rankings, which indicates a clear superiority in such scenario. This behavior can be explained by the use of neighboring solutions to learn the CMA-ES model, which can be advantageous when using few, evenly-spaced solutions.

7.4.2.4 Summary of the results

The use of multiple CMA-ES models simultaneously during a multi-objective search is very important, since it allows each model to focus in a smaller area of the search space. By allowing each model to focus in a small region of the search space, specialized models can be created, which approximate them to the single-objective approach that they were originally proposed to handle.

As far as we know, this is the first Pareto based MO-CMA-ES to employ rank mu update of multiple models. Moreover, we created two versions of this algorithm, one of them using a limited population and the other using an unbounded population.

To evaluate the behavior of our proposed algorithms, as well as to compare them to a classical MOEA approach, we conducted two experimental studies on a set of well-known benchmark problems using two and three objective functions. The results were evaluated using two quality indicators and the values measured by them were submitted to statistical tests to look for statistically significant differences.

Our results indicate that, in most of the problems studied, our proposed MO-CMA-ES variants are able to outperform the classical IBEA algorithm. Moreover, the variants using unbounded population outperform the other variants in most of the problems. When we remove these unbounded variants from the comparison, the MO-CMA-ES$_{CD}$-rankMu presents very good results, outperforming its rank one counterpart in almost all problems, and presenting better results than IBEA in most of the problems.

7.4.3 Empirical studies involving the dominance and decomposition based MO-CMA-ES

This section presents the three empirical studies conducted involving the MOEA/DD-CMA. First we present the experimental setup of these studies in Section 7.4.3.1. The first empirical study is conducted to assess the performance of the new MOEA/DD-CMA and to compare it with MOEA/D-CMA and is presented in Section 7.4.3.2. In Section 7.4.3.3, a second experimental study is carried out involving eight different neighborhood schemes, where they are compared using the MOEA/DD-CMA framework. Finally a third experimental study is performed and presented in Section 7.4.3.4, where we compare the performance of MOEA/DD-CMA to the bounded population variants of MO-CMA-ES-rankOne and MO-CMA-ES-rankMu.

7.4.3.1 Experimental setup

In the first two experimental studies, we used the well known DTLZ (Deb et al., 2005) and WFG (Huband et al., 2006) families of benchmark problems. The parameters used in these studies are a combination of those used by Li et al. (2015) and Zapotecas-Martínez et al. (2015), and are summarized in Table 7.32. The stop criterion is the number of generations, set according to the maximum number of generations used by Li et al. (2015). The number of solutions generated per subproblem is set according to the number of decision variables, as defined by Zapotecas-Martínez et al. (2015) as $\lambda = 4 + \lfloor 3 \ln n \rfloor$.

Table 7.32: General parameters used in the first experiments.

Parameter	m=2	m=3	m=5	m=8	m=10	m=15
Iterations	500	1000	1000	1500	2000	3000
Weight vectors	100	91	210	156	275	135
λ for DTLZ1	9	9	10	11	11	12
λ for DTLZ 2-4	11	11	11	12	12	13
λ for WFG	13	13	13	14	14	15

The weight vectors for the decomposition variants in all experimental studies are generated in a two layer weight vector generation method, as presented by Li et al. (2015) when the number of objectives is higher than five. The neighborhood size was set to $T = 20$ for all problems and objective numbers. The probability used to select solutions in the neighborhood was set to $\delta = 0.9$. A polynomial mutation operator is applied on 15% of the solutions with probability $p_m = 1/n$ and distribution index of $n_m = 20$ in order to increase the diversity and avoid local optima. Since we employed a mutation in the MOEA/DD-CMA, we also included it in the MOEA/D-CMA to make a fair comparison. The default values for the initialization of the CMA-ES parameters were already introduced along Section 3.6.2.

In the last experimental study, were we compare MOEA/DD-CMA to the MO-CMA-ES$_{CD}$-rankOne and the MO-CMA-ES$_{CD}$-rankMu variants, we only use the DTLZ family of problems. Moreover, we set as stopping criterion 50000 function evaluations for all algorithms. Due to the smaller budget, we set the MOEA/DD-CMA to sample only one solution per subproblem at each generation. For the MO-CMA-ES variants we set a population size of 100 solutions.

7.4.3.2 Comparison between MOEA/D-CMA and MOEA/DD-CMA

This section presents the results obtained in the first experimental study, where we compare our proposed MOEA/DD-CMA to the MOEA/D-CMA algorithm available on the literature. These results are presented through Tables 7.33 and 7.34 containing the IGD_p and

Hypervolume results for the DTLZ problems, respectively, and through Tables 7.36 and 7.37 containing the IGD_p and Hypervolume results for the WFG problems, respectively. In these tables, the first column represents the number of objectives, the second column represents the algorithm. The other columns show the mean ranks of the results obtained on 30 independent runs of the algorithms.

Table 7.33: Mean ranks of the IGD_p as used in the Kruskal-Wallis test for the DTLZ problems. Final ranks, presented in parentheses assigned according to mean ranks.

Obj.	Algorithms	DTLZ1	DTLZ2	DTLZ3	DTLZ4
2	MOEA/D-CMA	35.27 (2.00)	**29.33 (1.50)**	**34.10 (1.50)**	**26.57 (1.50)**
	MOEA/DD-CMA	**25.73 (1.00)**	**31.67 (1.50)**	**26.90 (1.50)**	**34.43 (1.50)**
3	MOEA/D-CMA	**32.63 (1.50)**	45.50 (2.00)	**31.40 (1.50)**	37.20 (2.00)
	MOEA/DD-CMA	**28.37 (1.50)**	**15.50 (1.00)**	**29.60 (1.50)**	**23.80 (1.00)**
5	MOEA/D-CMA	**34.87 (1.50)**	38.67 (2.00)	**32.93 (1.50)**	**32.67 (1.50)**
	MOEA/DD-CMA	**26.13 (1.50)**	**22.33 (1.00)**	**28.07 (1.50)**	**28.33 (1.50)**
8	MOEA/D-CMA	**29.60 (1.50)**	41.10 (2.00)	40.17 (2.00)	**29.83 (1.50)**
	MOEA/DD-CMA	**31.40 (1.50)**	**19.90 (1.00)**	**20.83 (1.00)**	**31.17 (1.50)**
10	MOEA/D-CMA	**28.23 (1.50)**	**33.20 (1.50)**	**32.67 (1.50)**	**34.30 (1.50)**
	MOEA/DD-CMA	**32.77 (1.50)**	**27.80 (1.50)**	**28.33 (1.50)**	**26.70 (1.50)**
15	MOEA/D-CMA	**31.00 (1.50)**	**32.93 (1.50)**	**29.27 (1.50)**	40.40 (2.00)
	MOEA/DD-CMA	**30.00 (1.50)**	**28.07 (1.50)**	**31.73 (1.50)**	**20.60 (1.00)**

Table 7.34: Mean ranks of the Hypervolume as used in the Kruskal-Wallis test for the DTLZ problems. Final ranks, presented in parentheses assigned according to mean ranks.

Obj.	Algorithms	DTLZ1	DTLZ2	DTLZ3	DTLZ4
2	MOEA/D-CMA	36.73 (2.00)	**32.13 (1.50)**	37.40 (2.00)	35.80 (2.00)
	MOEA/DD-CMA	**24.27 (1.00)**	**28.87 (1.50)**	**23.60 (1.00)**	**25.20 (1.00)**
3	MOEA/D-CMA	**34.27 (1.50)**	45.47 (2.00)	**33.27 (1.50)**	40.13 (2.00)
	MOEA/DD-CMA	**26.73 (1.50)**	**15.53 (1.00)**	**27.73 (1.50)**	**20.87 (1.00)**
5	MOEA/D-CMA	37.40 (2.00)	**24.93 (1.00)**	**30.50 (1.50)**	**22.53 (1.00)**
	MOEA/DD-CMA	**23.60 (1.00)**	36.07 (2.00)	**30.50 (1.50)**	38.47 (2.00)
8	MOEA/D-CMA	39.67 (2.00)	**34.20 (1.50)**	37.67 (2.00)	**15.63 (1.00)**
	MOEA/DD-CMA	**21.33 (1.00)**	**26.80 (1.50)**	**23.33 (1.00)**	45.37 (2.00)
10	MOEA/D-CMA	41.70 (2.00)	**32.20 (1.50)**	39.93 (2.00)	**16.73 (1.00)**
	MOEA/DD-CMA	**19.30 (1.00)**	**28.80 (1.50)**	**21.07 (1.00)**	44.27 (2.00)
15	MOEA/D-CMA	**29.03 (1.50)**	**34.73 (1.50)**	**30.50 (1.50)**	40.57 (2.00)
	MOEA/DD-CMA	**31.97 (1.50)**	**26.27 (1.50)**	**30.50 (1.50)**	**20.43 (1.00)**

In this analysis, first we consider the DTLZ problems. In this case, when observing the IGD_p results, in most cases the algorithms do not present statistically significant differences. However in the few instances (problem/objective number) where differences were found, in all of them the MOEA/DD-CMA had the best results. When considering the Hypervolume, the differences become clearer, however, overall the MOEA/DD-CMA achieves better performance than MOEA/D-CMA. More specifically, MOEA/DD-CMA outperformed MOEA/D-CMA on all instances of problems DTLZ1 and DTLZ3, these are multimodal problems where CMA-ES usually fails to approach the true Pareto front, which means that in these very hard problems the selection mechanism of MOEA/DD made a significant contribution to the algorithm. DTLZ2 is an easier problem that poses no challenge to the algorithms, in this case MOEA/D-CMA only outperformed MOEA/DD-CMA for five objectives. DTLZ4 is a problem that challenges the ability of the optimizers in obtaining a diverse set of solutions. In this case MOEA/D-CMA outperformed MOEA/DD-CMA for five, eight and ten objectives, this poor performance of MOEA/DD-CMA on higher numbers of objectives (except for fifteen) can be explained by the update mechanism of MOEA/DD, which introduces diversity in the search, hence the convergence characteristic of the algorithm is decreased. In an additional convergence analysis using the GD_p indicator, presented in Table 7.35, we can identify that in these particular instances MOEA/D-CMA outperformed MOEA/DD-CMA.

Table 7.35: Mean ranks of the GD_p as used in the Kruskal-Wallis test for the DTLZ problems. Final ranks, presented in parentheses assigned according to mean ranks.

Obj.	Algorithms	DTLZ1	DTLZ2	DTLZ3	DTLZ4
2	MOEA/D-CMA	35.27 (2.00)	**29.27 (1.50)**	36.70 (2.00)	**25.70 (1.00)**
	CMA-MOEA/DD	**25.73 (1.00)**	**31.73 (1.50)**	**24.30 (1.00)**	35.30 (2.00)
3	MOEA/D-CMA	**33.83 (1.50)**	45.50 (2.00)	**33.57 (1.50)**	38.30 (2.00)
	CMA-MOEA/DD	**27.17 (1.50)**	**15.50 (1.00)**	**27.43 (1.50)**	**22.70 (1.00)**
5	MOEA/D-CMA	**34.03 (1.50)**	39.97 (2.00)	**30.33 (1.50)**	**20.20 (1.00)**
	CMA-MOEA/DD	**26.97 (1.50)**	**21.03 (1.00)**	**30.67 (1.50)**	40.80 (2.00)
8	MOEA/D-CMA	**30.50 (1.50)**	38.17 (2.00)	**23.50 (1.00)**	**16.50 (1.00)**
	CMA-MOEA/DD	**30.50 (1.50)**	**22.83 (1.00)**	37.50 (2.00)	44.50 (2.00)
10	MOEA/D-CMA	**25.63 (1.00)**	**33.43 (1.50)**	**17.40 (1.00)**	**15.63 (1.00)**
	CMA-MOEA/DD	35.37 (2.00)	**27.57 (1.50)**	43.60 (2.00)	45.37 (2.00)
15	MOEA/D-CMA	**29.20 (1.50)**	**30.87 (1.50)**	**15.50 (1.00)**	**26.93 (1.50)**
	CMA-MOEA/DD	**31.80 (1.50)**	**30.13 (1.50)**	45.50 (2.00)	**34.07 (1.50)**

Table 7.36: Mean ranks of the IGD_p as used in the Kruskal-Wallis test for the WFG problems. Final ranks, presented in parentheses assigned according to mean ranks.

Obj.	Algorithms	WFG1	WFG2	WFG3	WFG4	WFG5	WFG6	WFG7	WFG8	WFG9
2	MOEA/D-CMA	45.50 (2.00)	45.50 (2.00)	45.50 (2.00)	43.67 (2.00)	45.50 (2.00)	45.50 (2.00)	45.50 (2.00)	45.27 (2.00)	45.37 (2.00)
	MOEA/DD-CMA	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**17.33 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**15.73 (1.00)**	**15.63 (1.00)**
3	MOEA/D-CMA	45.50 (2.00)	45.50 (2.00)	45.50 (2.00)	45.50 (2.00)	45.50 (2.00)	45.50 (2.00)	45.50 (2.00)	38.63 (2.00)	45.50 (2.00)
	MOEA/DD-CMA	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**22.37 (1.00)**	**15.50 (1.00)**
5	MOEA/D-CMA	45.50 (2.00)	45.50 (2.00)	43.93 (2.00)	45.50 (2.00)	45.50 (2.00)	45.50 (2.00)	45.50 (2.00)	35.20 (2.00)	42.37 (2.00)
	MOEA/DD-CMA	**15.50 (1.00)**	**15.50 (1.00)**	**17.07 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**25.80 (1.00)**	**18.63 (1.00)**
8	MOEA/D-CMA	45.50 (2.00)	41.07 (2.00)	41.27 (2.00)	45.50 (2.00)	**15.73 (1.00)**	**15.50 (1.00)**	**18.57 (1.00)**	**18.07 (1.00)**	40.83 (2.00)
	MOEA/DD-CMA	**15.50 (1.00)**	**19.93 (1.00)**	**19.73 (1.00)**	**15.50 (1.00)**	45.27 (2.00)	45.50 (2.00)	42.43 (2.00)	42.93 (2.00)	**20.17 (1.00)**
10	MOEA/D-CMA	45.50 (2.00)	43.17 (2.00)	**34.03 (1.50)**	45.50 (2.00)	**15.50 (1.00)**	**15.50 (1.00)**	**15.53 (1.00)**	**16.50 (1.00)**	45.50 (2.00)
	MOEA/DD-CMA	**15.50 (1.00)**	**17.83 (1.00)**	**26.97 (1.50)**	**15.50 (1.00)**	45.50 (2.00)	45.50 (2.00)	45.47 (2.00)	44.50 (2.00)	**15.50 (1.00)**
15	MOEA/D-CMA	45.50 (2.00)	**25.20 (1.00)**	**30.90 (1.50)**	45.07 (2.00)	**21.47 (1.00)**	**17.53 (1.00)**	45.37 (2.00)	**24.93 (1.00)**	45.50 (2.00)
	MOEA/DD-CMA	**15.50 (1.00)**	35.80 (2.00)	**30.10 (1.50)**	**15.93 (1.00)**	39.53 (2.00)	43.47 (2.00)	**15.63 (1.00)**	36.07 (2.00)	**15.50 (1.00)**

Table 7.37: Mean ranks of the Hypervolume as used in the Kruskal-Wallis test for the WFG problems. Final ranks, presented in parentheses assigned according to mean ranks.

Obj.	Algorithms	WFG1	WFG2	WFG3	WFG4	WFG5	WFG6	WFG7	WFG8	WFG9
2	MOEA/D-CMA	45.50 (2.00)	44.77 (2.00)	45.50 (2.00)	43.67 (2.00)	45.50 (2.00)	45.50 (2.00)	45.50 (2.00)	43.33 (2.00)	40.53 (2.00)
	MOEA/DD-CMA	**15.50 (1.00)**	**16.23 (1.00)**	**15.50 (1.00)**	**17.33 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**17.67 (1.00)**	**20.47 (1.00)**
3	MOEA/D-CMA	45.50 (2.00)	45.50 (2.00)	45.50 (2.00)	45.50 (2.00)	45.50 (2.00)	45.50 (2.00)	45.50 (2.00)	40.50 (2.00)	45.50 (2.00)
	MOEA/DD-CMA	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**20.50 (1.00)**	**15.50 (1.00)**
5	MOEA/D-CMA	45.50 (2.00)	45.50 (2.00)	44.77 (2.00)	45.50 (2.00)	45.50 (2.00)	45.50 (2.00)	45.50 (2.00)	36.30 (2.00)	45.50 (2.00)
	MOEA/DD-CMA	**15.50 (1.00)**	**15.50 (1.00)**	**16.23 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**24.70 (1.00)**	**15.50 (1.00)**
8	MOEA/D-CMA	45.50 (2.00)	42.13 (2.00)	45.50 (2.00)	45.50 (2.00)	**18.40 (1.00)**	42.27 (2.00)	**22.70 (1.00)**	43.23 (2.00)	45.37 (2.00)
	MOEA/DD-CMA	**15.50 (1.00)**	**18.87 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	42.60 (2.00)	**18.73 (1.00)**	38.30 (2.00)	**17.77 (1.00)**	**15.63 (1.00)**
10	MOEA/D-CMA	45.50 (2.00)	43.47 (2.00)	45.30 (2.00)	45.50 (2.00)	**15.50 (1.00)**	**20.40 (1.00)**	**16.47 (1.00)**	34.93 (2.00)	45.50 (2.00)
	MOEA/DD-CMA	**15.50 (1.00)**	**17.53 (1.00)**	**15.70 (1.00)**	**15.50 (1.00)**	45.50 (2.00)	40.60 (2.00)	44.53 (2.00)	**26.07 (1.00)**	**15.50 (1.00)**
15	MOEA/D-CMA	**30.50 (1.50)**	37.15 (2.00)	**30.50 (1.50)**	**30.50 (1.50)**	**30.50 (1.50)**	**30.50 (1.50)**	**30.50 (1.50)**	**30.50 (1.50)**	**30.50 (1.50)**
	MOEA/DD-CMA	**30.50 (1.50)**	**23.85 (1.00)**	**30.50 (1.50)**	**30.50 (1.50)**	**30.50 (1.50)**	**30.50 (1.50)**	**30.50 (1.50)**	**30.50 (1.50)**	**30.50 (1.50)**

Following, we present the analysis considering the WFG problems. When observing the IGD_p results, for two, three and five objectives, MOEA/DD-CMA significantly outperforms MOEA/D-CMA in all problems. MOEA/D-CMA begins being competitive from eight objectives onwards, where it performs particularly well on problems WFG5, WFG6, WFG7 and WFG8. When considering the Hypervolume, the results are similar to those measured by IGD_p, where MOEA/DD-CMA performs best in all problems for two to five objectives, however for eight objectives it is outperformed by MOEA/D-CMA on WFG5 and WFG7, while for ten objectives it is outperformed on WFG5, WFG6 and WFG7. The results obtained for fifteen objectives are all equal in most problems. This happens because we are using the approximate hypervolume and the results were so similar that the approximate version could not identify any difference.

Summarizing the results obtained using all problems and measured with all metrics, MOEA/DD-CMA presented better results than MOEA/D-CMA, especially for two and three objectives, where the selection pressure introduced by the dominance relation is higher. When the number of objectives increase, MOEA/DD-CMA still has overall better performance, however MOEA/D-CMA starts to be competitive.

7.4.3.3 Changing the neighborhood relation of MOEA/DD-CMA

This section presents the experimental study conducted to investigate the behavior of the four approaches proposed for dynamic recalculating the neighborhood relation among the subproblems at each iteration of the algorithm. Moreover, we compare the proposed approaches to three control methods to ensure that the differences found are not due to other external factors. Furthermore, these seven variants are compared to the original neighborhood relation used in MOEA/D.

First we conduct an analysis of the results obtained for the DTLZ problems. These results are presented in Tables 7.38 and 7.40 featuring the ranks of the IGD_p and hypervolume values. A summarized view of these results are presented in Tables 7.39 and 7.41 for the IGD_p and hypervolume respectively.

Considering the IGD_p results, in general, we can state that the approaches $N0$ and $N5$ usually appear among the best. Moreover, the approach $N3$ performs well for problems up to five objectives, however for eight objectives onwards its performance deteriorates. Other than this, it is very hard to recognize patterns. Regarding the summarized table, we can see that the original MOEA/D approach $N0$ outperforms the others with statistically significant differences.

Regarding the hypervolume results, again we can observe the approaches $N0$, $N3$ and $N5$ often achieving the best results, especially from three through eight objectives. However, as with the IGD, it is hard to extract additional patterns. The summarized table indicates the control method $N5$ (Euclidean distance between best solutions) as the overall best approach for the problems investigated. The original MOEA/D approach appears in second.

Now we analyze the results obtained when optimizing the WFG problems. These results are presented in Tables 7.42 and 7.44 containing the ranks of the IGD_p and hypervolume values. A summarized view of these results are presented in Tables 7.43 and 7.45 for the IGD_p and hypervolume respectively.

Considering the general IGD_p results, we are not able to recognize any patterns. The results are so similar, that in nine instances (problem/objective number) no statistically significant differences were found between any approaches. When considering the overall results, the approach $N2$ (likelihood given by model to the best solution of neighbors) performed the best, however it appears with a rank four, which means it statistically tied with all algorithms, except for $N5$.

Table 7.38: Mean ranks of the IGD_p as used in the Kruskal-Wallis test for the DTLZ problems. Final ranks, presented in parentheses assigned according to mean ranks.

Obj.	Algorithms	DTLZ1	DTLZ2	DTLZ3	DTLZ4
2	N0	87.17 (3.50)	**48.73 (2.00)**	133.67 (5.00)	**73.57 (3.00)**
	N1	152.43 (5.50)	169.17 (6.00)	112.20 (4.50)	172.27 (7.00)
	N2	138.70 (5.00)	156.77 (6.00)	131.33 (5.00)	175.47 (7.00)
	N3	**50.30 (1.50)**	**67.60 (2.00)**	**74.00 (2.50)**	94.00 (3.50)
	N4	152.17 (5.50)	152.13 (6.00)	110.33 (4.50)	111.50 (3.50)
	N5	125.80 (5.00)	**53.90 (2.00)**	126.43 (4.50)	103.23 (3.50)
	N6	132.07 (5.00)	173.53 (6.00)	136.63 (5.00)	131.40 (5.00)
	N7	125.37 (5.00)	142.17 (6.00)	139.40 (5.00)	102.57 (3.50)
3	N0	**65.97 (1.50)**	**49.10 (2.00)**	**133.90 (4.50)**	**41.77 (2.50)**
	N1	166.70 (5.50)	210.00 (7.00)	**99.87 (4.50)**	197.67 (6.50)
	N2	161.20 (5.50)	184.73 (6.50)	**137.77 (4.50)**	182.03 (6.50)
	N3	**35.00 (1.50)**	**50.90 (2.00)**	**90.53 (4.50)**	146.70 (6.00)
	N4	138.20 (5.50)	110.57 (5.00)	**104.03 (4.50)**	165.13 (6.50)
	N5	126.03 (5.50)	**43.23 (2.00)**	**136.20 (4.50)**	**84.73 (2.50)**
	N6	134.80 (5.50)	163.23 (6.00)	**137.50 (4.50)**	93.30 (3.00)
	N7	136.10 (5.50)	152.23 (5.50)	**124.20 (4.50)**	**52.67 (2.50)**
5	N0	91.77 (3.50)	**35.97 (2.00)**	**111.67 (4.50)**	**47.70 (2.00)**
	N1	129.27 (5.00)	218.07 (8.00)	**113.30 (4.50)**	156.97 (6.00)
	N2	153.70 (5.50)	150.40 (5.50)	**125.73 (4.50)**	128.50 (5.50)
	N3	**61.43 (2.50)**	81.13 (2.50)	**108.13 (4.50)**	92.10 (2.50)
	N4	113.93 (4.50)	139.77 (5.50)	**102.40 (4.50)**	151.13 (6.00)
	N5	107.03 (4.50)	73.20 (2.50)	**123.10 (4.50)**	**46.07 (2.00)**
	N6	161.23 (5.50)	111.83 (4.50)	**146.03 (4.50)**	166.17 (6.00)
	N7	145.63 (5.00)	153.63 (5.50)	**133.63 (4.50)**	175.37 (6.00)
8	N0	**78.53 (3.00)**	217.23 (7.50)	112.27 (4.50)	**50.67 (2.00)**
	N1	139.90 (5.50)	117.30 (5.00)	122.98 (4.50)	151.80 (5.50)
	N2	127.43 (4.50)	**60.17 (1.50)**	149.38 (5.00)	181.50 (7.00)
	N3	103.73 (4.00)	139.97 (5.00)	132.03 (4.50)	120.57 (4.50)
	N4	122.90 (4.50)	163.53 (5.50)	112.02 (4.50)	90.37 (3.00)
	N5	**73.40 (3.00)**	118.73 (5.00)	**85.70 (4.00)**	100.30 (4.00)
	N6	175.40 (6.00)	**31.17 (1.50)**	138.22 (4.50)	116.43 (4.50)
	N7	142.70 (5.50)	115.90 (5.00)	111.40 (4.50)	152.37 (5.50)
10	N0	**63.70 (2.00)**	215.53 (7.50)	100.00 (4.00)	**36.43 (2.00)**
	N1	129.78 (5.00)	115.77 (4.00)	112.60 (4.50)	127.77 (5.00)
	N2	123.77 (4.50)	79.30 (3.00)	141.73 (5.00)	136.83 (5.50)
	N3	101.78 (4.00)	125.93 (4.50)	111.53 (4.50)	171.03 (6.00)
	N4	126.75 (4.50)	138.10 (5.50)	100.00 (4.00)	**41.20 (2.00)**
	N5	87.43 (3.50)	179.53 (6.50)	**87.30 (3.00)**	79.50 (2.50)
	N6	182.52 (7.00)	**29.40 (2.00)**	162.30 (6.00)	194.70 (7.00)
	N7	148.27 (5.50)	80.43 (3.00)	148.53 (5.00)	176.53 (6.00)
15	N0	**93.63 (3.50)**	**20.63 (1.50)**	**120.43 (4.50)**	**68.20 (2.50)**
	N1	**89.47 (3.50)**	191.77 (7.00)	**92.10 (4.50)**	157.67 (6.00)
	N2	**122.17 (3.50)**	133.33 (5.00)	**130.93 (4.50)**	146.70 (5.00)
	N3	**69.97 (3.50)**	175.30 (5.50)	**109.70 (4.50)**	149.60 (5.00)
	N4	**92.80 (3.50)**	121.90 (5.00)	**121.93 (4.50)**	99.37 (4.00)
	N5	**122.30 (3.50)**	**46.60 (1.50)**	**110.23 (4.50)**	102.90 (4.00)
	N6	195.07 (7.50)	140.77 (5.50)	**136.87 (4.50)**	123.43 (5.00)
	N7	178.60 (7.50)	133.70 (5.00)	**141.80 (4.50)**	116.13 (4.50)

Table 7.39: Overall ranks of IGD_p as used in the Friedman test for all the DTLZ problems and numbers of objectives. Final ranks, presented in parentheses assigned according to the ranks.

N0	N1	N2	N3	N4	N5	N6	N7
71.0 (2.5)	139.0 (6.0)	139.0 (6.0)	77.0 (3.0)	97.0 (4.5)	76.0 (3.0)	142.0 (6.0)	123.0 (5.0)

The hypervolume results are similar to those of IGD, in the sense that no clear patterns can be extracted from them. When looking at the overall results, again $N2$ achieved the best results, although it statistically tied with all approaches, except for $N1$ and $N5$.

In a summarized analysis of the results, an interesting pattern emerges, while for the DTLZ family of problems the approaches $N0$, $N5$ and $N3$ performed well, for the WFG family they performed very poorly, with the $N5$ achieving the worst general rankings considering both indicators. Similarly, for the WFG problems, $N2$ outperformed the other approaches, however for the DTLZ problems, considering the IGD, $N2$ had the second worst ranking, tied with $N1$, and considering the hypervolume, $N2$ had the absolute worst ranking. This summarized analysis

Table 7.40: Mean ranks of the hypervolume as used in the Kruskal-Wallis test for the DTLZ problems. Final ranks, presented in parentheses assigned according to mean ranks.

Obj.	Algorithms	DTLZ1	DTLZ2	DTLZ3	DTLZ4
2	N0	81.77 (3.00)	98.13 (3.50)	130.67 (5.00)	**39.57 (2.00)**
	N1	157.30 (5.50)	199.23 (7.00)	102.67 (4.00)	211.43 (7.50)
	N2	146.70 (5.50)	202.37 (7.00)	125.70 (5.00)	204.20 (7.50)
	N3	**44.33 (1.50)**	175.30 (7.00)	**58.27 (2.00)**	140.07 (5.00)
	N4	163.87 (5.50)	71.00 (3.00)	111.23 (4.50)	108.17 (4.00)
	N5	118.77 (5.00)	76.63 (3.50)	135.00 (5.00)	66.20 (2.50)
	N6	128.13 (5.00)	120.57 (3.50)	157.97 (5.50)	132.27 (5.00)
	N7	123.13 (5.00)	**20.77 (1.50)**	142.50 (5.00)	62.10 (2.50)
3	N0	57.90 (2.00)	**49.37 (2.00)**	103.90 (4.50)	**34.23 (1.50)**
	N1	164.33 (5.50)	217.37 (7.50)	103.80 (4.50)	189.97 (6.50)
	N2	161.03 (5.50)	200.90 (7.00)	136.67 (4.50)	169.03 (6.50)
	N3	**32.33 (1.50)**	**49.13 (2.00)**	**91.37 (3.50)**	149.63 (6.50)
	N4	145.27 (5.50)	109.03 (5.00)	109.20 (4.50)	174.43 (6.50)
	N5	111.47 (5.00)	**38.37 (2.00)**	120.53 (4.50)	95.30 (3.00)
	N6	148.93 (5.50)	155.57 (5.50)	150.40 (5.00)	90.47 (3.00)
	N7	142.73 (5.50)	144.27 (5.00)	148.13 (5.00)	60.93 (2.50)
5	N0	**68.57 (2.00)**	**39.63 (2.00)**	**97.27 (3.50)**	59.20 (2.00)
	N1	130.90 (5.00)	221.83 (7.50)	113.93 (4.50)	198.90 (7.50)
	N2	145.30 (5.50)	181.53 (6.50)	125.40 (4.50)	148.73 (5.50)
	N3	**43.40 (2.00)**	76.40 (2.50)	103.27 (4.00)	**50.73 (1.50)**
	N4	125.97 (5.00)	115.20 (4.50)	107.17 (4.50)	142.67 (5.00)
	N5	91.43 (3.50)	**37.97 (2.00)**	102.13 (4.00)	105.80 (4.50)
	N6	197.27 (7.00)	141.67 (5.50)	155.93 (5.00)	119.93 (5.00)
	N7	161.17 (6.00)	149.77 (5.50)	158.90 (6.00)	138.03 (5.00)
8	N0	**81.00 (3.50)**	197.87 (7.00)	90.63 (4.00)	85.43 (3.00)
	N1	130.77 (4.00)	170.00 (6.50)	**81.57 (3.50)**	124.07 (5.00)
	N2	119.23 (4.00)	104.53 (3.50)	134.27 (4.50)	148.10 (5.50)
	N3	**92.23 (3.50)**	**66.97 (2.50)**	133.30 (4.50)	118.53 (4.50)
	N4	109.23 (4.00)	158.20 (6.00)	130.53 (4.50)	**68.70 (2.50)**
	N5	**84.20 (3.50)**	**60.13 (2.50)**	**85.60 (3.50)**	115.27 (4.50)
	N6	192.50 (7.50)	**71.10 (2.50)**	163.17 (6.00)	143.07 (5.50)
	N7	154.83 (6.00)	135.20 (5.50)	144.93 (5.50)	160.83 (5.50)
10	N0	**97.03 (4.00)**	219.13 (7.50)	87.22 (4.00)	65.47 (3.00)
	N1	**117.43 (4.00)**	**69.80 (3.00)**	116.03 (4.00)	110.50 (3.00)
	N2	**129.07 (4.00)**	100.23 (3.50)	126.02 (4.50)	117.90 (4.50)
	N3	**92.00 (4.00)**	**92.90 (3.00)**	198.13 (8.00)	169.40 (6.50)
	N4	**101.08 (4.00)**	174.90 (7.00)	113.30 (4.00)	82.97 (3.00)
	N5	**98.57 (4.00)**	151.30 (6.00)	**62.38 (2.50)**	**57.50 (2.50)**
	N6	187.70 (7.50)	**61.23 (3.00)**	138.38 (4.50)	192.70 (7.00)
	N7	141.12 (4.50)	**94.50 (3.00)**	122.53 (4.50)	167.57 (6.50)
15	N0	104.20 (4.50)	**30.35 (1.50)**	**120.50 (4.50)**	166.32 (6.50)
	N1	100.33 (4.00)	193.33 (7.00)	**120.50 (4.50)**	143.13 (5.50)
	N2	145.02 (5.00)	136.73 (5.00)	**120.50 (4.50)**	111.62 (3.50)
	N3	**75.47 (3.00)**	157.40 (5.50)	**120.50 (4.50)**	**71.77 (3.00)**
	N4	103.30 (4.50)	126.37 (5.00)	**120.50 (4.50)**	166.87 (6.50)
	N5	126.83 (4.50)	**38.38 (1.50)**	**120.50 (4.50)**	123.18 (4.50)
	N6	156.80 (5.50)	143.33 (5.50)	**120.50 (4.50)**	98.02 (3.50)
	N7	152.05 (5.00)	138.10 (5.00)	**120.50 (4.50)**	**83.10 (3.00)**

Table 7.41: Overall ranks of hypervolume as used in the Friedman test for all the DTLZ problems and numbers of objectives. Final ranks, presented in parentheses assigned according to the ranks.

N0	N1	N2	N3	N4	N5	N6	N7
76.5 (3.0)	129.5 (5.5)	143.5 (6.0)	81.5 (3.5)	104.5 (4.5)	**69.5 (2.5)**	137.5 (6.0)	121.5 (5.0)

reveals that the approach used for defining the neighborhood relations between the subproblems can be an important factor in the algorithms, however its selection is highly problem dependent.

Table 7.42: Mean ranks of the IGD_p as used in the Kruskal-Wallis test for the WFG problems. Final ranks, presented in parentheses assigned according to mean ranks.

Obj.	Algs.	WFG1	WFG2	WFG3	WFG4	WFG5	WFG6	WFG7	WFG8	WFG9
2	N0	112.90 (4.50)	**76.43 (3.00)**	**70.47 (2.50)**	47.47 (2.00)	47.70 (2.00)	49.83 (2.00)	**45.90 (2.50)**	**45.97 (2.00)**	**41.70 (2.00)**
	N1	136.00 (5.50)	157.47 (6.50)	186.10 (6.50)	198.53 (7.00)	207.30 (7.00)	210.17 (7.00)	199.83 (7.00)	209.30 (7.00)	182.07 (6.00)
	N2	**41.27 (2.00)**	**58.57 (3.00)**	**53.97 (2.50)**	93.10 (3.50)	104.63 (4.00)	110.20 (5.00)	**60.27 (2.50)**	**46.87 (2.00)**	131.53 (6.00)
	N3	79.47 (2.50)	**90.33 (3.00)**	**73.60 (2.50)**	143.43 (5.00)	**22.30 (1.50)**	**26.33 (1.50)**	115.83 (4.00)	102.97 (4.50)	**44.87 (2.00)**
	N4	**57.50 (2.00)**	185.30 (7.00)	**68.93 (2.50)**	**28.40 (1.50)**	78.27 (3.00)	89.37 (3.00)	64.70 (3.00)	**46.33 (2.00)**	**69.30 (2.00)**
	N5	201.00 (7.00)	192.47 (7.00)	191.57 (6.50)	115.30 (4.50)	138.50 (5.50)	162.57 (6.00)	90.80 (3.00)	191.03 (6.50)	183.93 (6.00)
	N6	173.27 (6.50)	**92.60 (3.00)**	157.37 (6.50)	156.23 (6.00)	186.90 (6.50)	162.17 (6.00)	188.47 (7.00)	152.17 (5.50)	165.00 (6.00)
	N7	162.60 (6.00)	110.83 (3.50)	162.00 (6.50)	181.53 (6.50)	178.40 (6.50)	153.37 (5.50)	198.20 (7.00)	169.37 (6.50)	145.60 (6.00)
3	N0	139.53 (5.00)	143.63 (5.50)	92.17 (3.50)	**59.23 (2.00)**	96.47 (3.50)	152.43 (5.50)	**30.60 (1.00)**	**48.63 (2.00)**	85.30 (3.50)
	N1	200.43 (7.50)	**48.33 (2.50)**	159.33 (6.00)	67.53 (2.50)	**82.63 (2.50)**	169.30 (6.50)	178.73 (7.00)	224.97 (7.50)	177.73 (6.50)
	N2	115.10 (3.50)	92.07 (3.50)	144.50 (5.00)	**65.60 (2.00)**	88.00 (3.00)	99.37 (3.50)	116.00 (4.00)	77.83 (2.50)	129.73 (5.00)
	N3	212.17 (7.50)	144.37 (5.50)	147.53 (5.50)	218.57 (8.00)	93.70 (3.50)	140.80 (5.50)	169.57 (6.50)	143.63 (5.50)	96.70 (4.00)
	N4	88.93 (3.50)	161.00 (6.50)	94.37 (4.00)	120.70 (5.00)	146.97 (5.50)	113.43 (4.00)	170.83 (7.00)	**34.60 (2.00)**	**66.20 (2.00)**
	N5	**76.67 (3.00)**	214.93 (7.50)	**71.03 (2.50)**	149.00 (5.50)	139.77 (5.00)	169.83 (6.50)	100.23 (3.50)	105.47 (4.50)	150.63 (5.50)
	N6	**62.37 (3.00)**	**80.50 (2.50)**	120.67 (4.50)	135.43 (5.50)	151.83 (6.50)	**54.23 (2.00)**	91.77 (3.50)	154.83 (5.50)	122.37 (4.50)
	N7	**68.80 (3.00)**	**79.17 (2.50)**	134.40 (5.00)	147.93 (5.50)	164.63 (6.50)	64.60 (2.50)	106.27 (3.50)	174.03 (6.50)	135.33 (5.00)
5	N0	120.23 (4.50)	125.93 (4.50)	108.37 (3.50)	177.40 (7.00)	**63.97 (2.50)**	161.10 (6.50)	**69.07 (3.50)**	72.67 (3.00)	76.13 (3.00)
	N1	184.67 (6.50)	**83.50 (3.50)**	**73.40 (2.50)**	**48.23 (2.00)**	**65.57 (2.50)**	200.40 (7.00)	213.10 (7.50)	197.63 (7.00)	193.13 (7.50)
	N2	134.17 (5.00)	**96.17 (3.50)**	**58.33 (2.50)**	189.73 (7.00)	**70.60 (2.50)**	**34.27 (2.00)**	**66.50 (3.50)**	73.90 (3.00)	179.30 (7.00)
	N3	216.17 (7.50)	152.13 (5.50)	153.23 (6.00)	103.50 (4.00)	143.87 (5.50)	95.67 (3.50)	**119.67 (3.50)**	126.43 (4.50)	114.70 (3.50)
	N4	143.63 (5.00)	107.23 (4.50)	**72.97 (2.50)**	99.00 (3.50)	**72.27 (2.50)**	119.67 (4.00)	**71.33 (3.50)**	**61.60 (2.50)**	**63.90 (2.50)**
	N5	92.23 (4.50)	160.60 (6.00)	135.50 (6.00)	135.77 (5.00)	138.93 (5.50)	204.07 (7.00)	187.37 (7.50)	**55.33 (2.50)**	130.83 (5.00)
	N6	**37.80 (1.50)**	104.50 (4.00)	178.67 (6.50)	115.80 (4.00)	202.80 (7.50)	65.47 (3.00)	**118.97 (3.50)**	179.20 (6.50)	118.50 (4.00)
	N7	**35.10 (1.50)**	133.93 (4.50)	183.53 (6.50)	94.57 (3.50)	206.00 (7.50)	83.37 (3.00)	**118.00 (3.50)**	197.23 (7.00)	87.50 (3.50)
8	N0	130.37 (5.00)	**133.80 (4.50)**	**118.97 (4.50)**	146.40 (5.00)	194.03 (6.50)	146.87 (5.50)	136.90 (5.00)	82.10 (3.00)	107.37 (4.50)
	N1	**40.53 (1.50)**	**116.30 (4.50)**	**105.37 (4.50)**	132.40 (4.50)	**53.03 (2.50)**	136.20 (5.00)	**48.37 (2.50)**	91.67 (3.00)	142.57 (4.50)
	N2	**46.67 (1.50)**	**129.53 (4.50)**	**112.73 (4.50)**	139.77 (5.00)	**65.37 (2.50)**	148.00 (5.50)	167.93 (6.50)	**69.93 (2.50)**	**96.30 (4.00)**
	N3	147.90 (5.50)	**113.47 (4.50)**	**129.20 (4.50)**	126.87 (4.50)	157.40 (6.50)	132.67 (5.00)	125.77 (4.50)	172.03 (6.50)	152.23 (5.50)
	N4	126.97 (5.00)	**105.20 (4.50)**	**122.53 (4.50)**	**81.20 (3.00)**	165.00 (6.50)	128.67 (5.00)	187.70 (7.00)	**41.87 (2.50)**	133.87 (4.50)
	N5	199.53 (7.50)	**149.10 (4.50)**	**125.70 (4.50)**	137.97 (5.00)	175.10 (6.50)	109.17 (4.50)	94.40 (3.50)	130.80 (5.00)	130.00 (4.50)
	N6	138.93 (5.00)	**97.60 (4.50)**	**124.20 (4.50)**	104.20 (4.50)	**77.87 (2.50)**	89.77 (3.50)	100.93 (3.50)	180.83 (6.50)	**95.40 (4.00)**
	N7	133.10 (5.00)	**119.00 (4.50)**	**125.30 (4.50)**	95.20 (4.50)	**76.20 (2.50)**	**72.67 (2.00)**	102.00 (3.50)	194.77 (7.00)	106.27 (4.50)
10	N0	188.40 (7.00)	**115.50 (4.50)**	**126.70 (4.50)**	170.97 (6.50)	206.57 (7.50)	115.50 (4.00)	151.43 (5.00)	112.23 (4.00)	114.10 (4.50)
	N1	**36.90 (2.00)**	**114.40 (4.50)**	**107.57 (4.50)**	**87.37 (4.00)**	**78.83 (3.00)**	**80.43 (3.50)**	**24.57 (1.00)**	**38.83 (1.50)**	159.53 (5.50)
	N2	71.70 (3.00)	**131.27 (4.50)**	**110.87 (4.50)**	127.27 (4.50)	117.83 (4.00)	120.87 (4.00)	118.90 (5.00)	**23.60 (1.50)**	155.60 (5.00)
	N3	165.67 (6.50)	**128.63 (4.50)**	**118.60 (4.50)**	**114.93 (4.00)**	121.07 (4.00)	187.10 (7.50)	160.20 (5.50)	198.70 (7.00)	**88.70 (3.50)**
	N4	80.77 (3.00)	**133.90 (4.50)**	**113.70 (4.50)**	**93.80 (4.00)**	136.20 (5.50)	140.63 (5.00)	160.10 (5.50)	96.57 (4.00)	116.73 (4.50)
	N5	186.80 (7.00)	**105.77 (4.50)**	**123.97 (4.50)**	132.80 (4.50)	163.27 (6.00)	129.63 (4.00)	98.90 (4.00)	127.53 (4.00)	122.13 (4.50)
	N6	124.13 (4.00)	**132.63 (4.50)**	**122.27 (4.50)**	**113.63 (4.00)**	**69.13 (3.00)**	88.00 (4.00)	119.27 (5.00)	182.17 (7.00)	101.77 (4.00)
	N7	109.63 (3.50)	**101.90 (4.50)**	**140.33 (4.50)**	123.23 (4.50)	**71.10 (3.00)**	101.83 (4.00)	130.63 (5.00)	184.37 (7.00)	105.43 (4.50)
15	N0	192.87 (7.00)	**120.77 (4.50)**	**135.75 (4.50)**	154.47 (5.50)	**123.77 (4.50)**	144.23 (5.50)	**125.40 (4.50)**	166.33 (5.50)	**118.23 (4.50)**
	N1	**44.67 (2.00)**	**116.70 (4.50)**	**116.73 (4.50)**	98.83 (4.00)	**143.20 (4.50)**	**72.23 (2.00)**	**90.77 (4.50)**	**32.23 (1.50)**	**117.83 (4.50)**
	N2	92.27 (3.00)	**136.73 (4.50)**	**118.35 (4.50)**	113.77 (4.50)	**103.97 (4.50)**	133.23 (5.00)	**124.15 (4.50)**	**35.73 (1.50)**	**129.60 (4.50)**
	N3	117.17 (4.00)	**104.73 (4.50)**	**132.88 (4.50)**	**66.87 (2.50)**	**115.53 (4.50)**	130.70 (5.00)	**112.62 (4.50)**	155.12 (5.50)	**118.90 (4.50)**
	N4	**39.13 (2.00)**	**127.23 (4.50)**	**114.27 (4.50)**	153.17 (5.00)	**144.03 (4.50)**	80.50 (3.00)	**135.78 (4.50)**	137.53 (5.50)	**116.23 (4.50)**
	N5	116.23 (4.00)	**110.63 (4.50)**	**121.55 (4.50)**	124.83 (5.00)	**103.53 (4.50)**	116.27 (4.50)	**130.08 (4.50)**	160.00 (5.50)	**116.40 (4.50)**
	N6	181.83 (7.00)	**122.77 (4.50)**	**112.43 (4.50)**	136.20 (5.00)	**104.10 (4.50)**	140.20 (5.50)	**135.53 (4.50)**	155.35 (5.50)	**113.43 (4.50)**
	N7	179.83 (7.00)	**124.43 (4.50)**	**112.03 (4.50)**	115.87 (4.50)	**125.87 (4.50)**	146.63 (5.50)	**109.67 (4.50)**	121.70 (5.50)	**133.37 (4.50)**

Table 7.43: Overall ranks of IGD_p as used in the Friedman test for all the WFG problems and numbers of objectives. Final ranks, presented in parentheses assigned according to the ranks.

N0	N1	N2	N3	N4	N5	N6	N7
242.0 (4.5)	244.0 (4.5)	**199.0 (4.0)**	262.0 (4.5)	216.0 (4.5)	287.0 (5.0)	237.0 (4.5)	257.0 (4.5)

Table 7.44: Mean ranks of the hypervolume as used in the Kruskal-Wallis test for the WFG problems. Final ranks, presented in parentheses assigned according to mean ranks.

Obj.	Algs.	WFG1	WFG2	WFG3	WFG4	WFG5	WFG6	WFG7	WFG8	WFG9
2	N0	117.83 (4.50)	76.47 (3.00)	**64.20 (2.50)**	**33.50 (1.50)**	50.50 (2.50)	59.70 (2.50)	**51.30 (2.50)**	**55.23 (2.50)**	**45.10 (2.00)**
	N1	141.43 (5.50)	158.50 (6.50)	182.70 (6.50)	208.40 (7.00)	207.50 (7.00)	209.30 (6.50)	204.43 (7.00)	202.13 (6.50)	160.30 (6.00)
	N2	**37.27 (2.00)**	**48.20 (2.50)**	**48.80 (2.50)**	95.33 (4.00)	104.53 (3.50)	90.67 (3.00)	**41.30 (2.50)**	**55.10 (2.50)**	143.93 (6.00)
	N3	70.23 (2.50)	100.00 (3.00)	**84.87 (2.50)**	141.87 (5.00)	**20.27 (1.50)**	**27.07 (1.50)**	120.10 (4.00)	**90.30 (2.50)**	**49.87 (2.00)**
	N4	**51.10 (2.00)**	176.70 (7.00)	**65.43 (2.50)**	**37.50 (1.50)**	76.60 (3.00)	86.23 (3.00)	71.33 (3.00)	**41.37 (2.50)**	**72.97 (2.00)**
	N5	202.63 (7.00)	206.47 (7.00)	194.73 (6.50)	100.43 (4.00)	137.13 (5.50)	172.10 (6.50)	93.47 (3.00)	203.30 (6.50)	182.30 (6.00)
	N6	170.03 (6.00)	88.10 (3.00)	160.97 (6.50)	160.40 (6.50)	186.73 (6.50)	163.67 (6.50)	186.40 (7.00)	151.63 (6.50)	174.50 (6.00)
	N7	173.47 (6.50)	109.57 (4.00)	162.30 (6.50)	186.57 (6.50)	180.73 (6.50)	155.27 (6.50)	195.67 (7.00)	164.93 (6.50)	135.03 (6.00)
3	N0	126.67 (5.00)	168.57 (6.00)	93.97 (4.00)	55.03 (2.50)	104.03 (4.50)	153.40 (5.50)	**30.33 (1.50)**	45.23 (2.50)	91.53 (4.00)
	N1	214.53 (7.50)	**49.47 (2.50)**	195.63 (7.50)	**32.97 (2.00)**	**80.57 (2.50)**	176.27 (7.00)	176.43 (6.50)	207.43 (7.00)	184.83 (7.50)
	N2	112.53 (4.00)	75.73 (3.00)	135.67 (4.50)	56.33 (2.50)	89.33 (3.00)	113.27 (4.50)	124.83 (4.50)	93.63 (3.50)	141.13 (5.00)
	N3	196.50 (7.50)	120.37 (4.50)	106.07 (4.00)	223.37 (8.00)	94.37 (4.00)	111.97 (4.00)	182.27 (7.00)	178.60 (6.50)	115.33 (4.00)
	N4	**55.83 (2.00)**	146.37 (5.50)	92.63 (4.00)	102.27 (3.00)	148.43 (5.50)	120.03 (4.50)	158.33 (6.00)	**35.63 (1.50)**	**66.47 (3.00)**
	N5	123.43 (4.50)	225.50 (8.00)	**71.57 (3.00)**	164.13 (6.00)	143.03 (5.00)	167.50 (6.00)	112.20 (4.00)	97.83 (3.50)	119.43 (4.00)
	N6	64.73 (2.50)	84.07 (3.00)	124.53 (4.00)	161.13 (6.00)	146.33 (5.50)	**54.93 (1.50)**	83.87 (3.00)	142.40 (5.00)	118.47 (4.00)
	N7	69.77 (3.00)	93.93 (3.50)	143.93 (5.00)	168.77 (6.00)	157.90 (6.00)	66.63 (3.00)	95.73 (3.50)	163.23 (6.50)	126.80 (4.50)
5	N0	122.70 (4.00)	105.07 (4.00)	**85.17 (3.00)**	143.50 (5.50)	201.67 (7.00)	174.80 (6.50)	163.53 (5.50)	**81.93 (2.50)**	136.77 (5.00)
	N1	106.13 (3.50)	102.23 (4.00)	123.40 (4.50)	**21.80 (1.50)**	**26.30 (1.50)**	172.50 (6.50)	223.73 (8.00)	182.23 (6.50)	77.67 (3.00)
	N2	134.77 (5.50)	**45.97 (1.50)**	**74.60 (3.00)**	60.43 (2.00)	93.17 (3.50)	114.93 (4.00)	87.97 (3.00)	**40.00 (2.50)**	**35.87 (1.50)**
	N3	217.33 (7.50)	196.30 (7.50)	139.83 (6.00)	137.07 (5.50)	113.17 (4.50)	69.50 (2.50)	116.67 (5.00)	137.13 (6.00)	199.53 (7.50)
	N4	71.60 (3.00)	111.93 (4.00)	**83.27 (3.00)**	80.47 (2.50)	141.03 (5.00)	143.70 (6.00)	143.43 (5.50)	**78.57 (2.50)**	190.33 (7.00)
	N5	180.70 (7.00)	213.33 (7.50)	109.67 (3.50)	201.13 (7.00)	72.50 (2.50)	190.33 (6.50)	165.87 (5.50)	**60.47 (2.50)**	93.30 (4.00)
	N6	68.73 (3.00)	82.83 (3.50)	176.30 (6.50)	150.73 (6.00)	152.00 (6.00)	**49.17 (2.00)**	**23.03 (1.50)**	179.93 (6.50)	125.07 (4.00)
	N7	**62.03 (2.50)**	106.33 (4.00)	171.77 (6.50)	168.87 (6.00)	164.17 (6.00)	**49.07 (2.00)**	39.77 (2.00)	203.73 (7.00)	105.47 (4.00)
8	N0	179.03 (6.00)	146.33 (5.50)	**91.37 (4.00)**	116.57 (4.50)	197.20 (6.50)	154.07 (5.50)	177.63 (7.00)	**60.43 (3.00)**	88.13 (3.50)
	N1	**29.17 (1.50)**	113.50 (4.50)	144.10 (4.50)	123.83 (4.50)	**90.87 (2.50)**	116.80 (5.00)	224.23 (7.50)	175.03 (7.00)	96.37 (3.50)
	N2	**54.60 (1.50)**	**84.40 (3.00)**	146.87 (5.00)	116.93 (4.50)	**44.77 (2.50)**	131.60 (5.50)	72.97 (3.00)	177.57 (7.00)	**56.10 (2.00)**
	N3	131.07 (5.50)	135.63 (4.50)	114.20 (4.50)	146.87 (5.00)	165.00 (6.50)	137.17 (5.50)	**31.67 (1.50)**	189.97 (7.00)	162.17 (6.00)
	N4	137.40 (5.50)	142.50 (5.00)	102.67 (4.50)	**84.67 (3.50)**	161.40 (6.50)	162.33 (5.50)	162.37 (6.00)	**109.97 (3.00)**	130.53 (5.00)
	N5	180.80 (6.00)	154.20 (6.00)	111.07 (4.50)	145.00 (5.00)	196.87 (6.50)	131.87 (5.50)	109.33 (4.00)	**102.87 (3.00)**	131.80 (5.00)
	N6	128.50 (5.50)	97.90 (4.00)	124.00 (4.50)	107.07 (4.50)	**52.13 (2.50)**	72.70 (2.00)	98.87 (3.50)	**69.80 (3.00)**	161.63 (6.00)
	N7	123.43 (4.50)	89.53 (3.50)	129.73 (4.50)	123.07 (4.50)	**55.77 (2.50)**	**57.47 (1.50)**	86.93 (3.50)	**78.37 (3.00)**	137.27 (5.00)
10	N0	204.17 (7.50)	161.23 (5.50)	104.70 (3.50)	**90.97 (3.50)**	199.67 (7.00)	135.50 (5.50)	182.87 (7.00)	**52.33 (3.00)**	114.20 (4.50)
	N1	**25.07 (1.50)**	84.07 (3.50)	175.73 (7.00)	**96.80 (3.50)**	**63.70 (3.00)**	113.03 (5.00)	218.53 (7.50)	215.23 (7.00)	103.37 (4.00)
	N2	70.70 (2.50)	122.53 (4.50)	184.03 (7.00)	114.50 (4.50)	**99.83 (3.00)**	123.33 (5.00)	71.23 (3.00)	183.57 (7.00)	**58.57 (1.50)**
	N3	125.43 (4.50)	111.03 (4.50)	144.00 (5.00)	112.07 (4.50)	154.73 (6.50)	187.93 (6.50)	**50.23 (2.00)**	185.67 (7.00)	179.93 (7.00)
	N4	102.07 (4.00)	137.93 (5.00)	98.90 (3.50)	137.60 (4.50)	107.20 (3.50)	167.10 (5.50)	147.83 (5.50)	**81.00 (3.00)**	128.63 (5.00)
	N5	186.83 (7.00)	157.90 (5.50)	100.60 (3.50)	**93.20 (3.50)**	194.23 (7.00)	151.83 (5.50)	112.90 (4.00)	**91.73 (3.00)**	121.93 (4.50)
	N6	134.83 (5.00)	**80.13 (3.00)**	**65.27 (3.00)**	158.27 (6.00)	**75.43 (3.00)**	**43.70 (1.50)**	107.33 (4.00)	**72.87 (3.00)**	141.50 (5.00)
	N7	114.90 (4.00)	109.17 (4.50)	90.77 (3.50)	160.60 (6.00)	**69.20 (3.00)**	**41.57 (1.50)**	73.07 (3.00)	**81.60 (3.00)**	115.87 (4.50)
15	N0	**120.50 (4.50)**	**111.30 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**
	N1	**120.50 (4.50)**	**134.37 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**
	N2	**120.50 (4.50)**	**99.23 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**
	N3	**120.50 (4.50)**	**145.40 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**
	N4	**120.50 (4.50)**	**137.90 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**
	N5	**120.50 (4.50)**	**125.08 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**
	N6	**120.50 (4.50)**	**96.07 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**
	N7	**120.50 (4.50)**	**114.65 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**	**120.50 (4.50)**

Table 7.45: Overall ranks of hypervolume as used in the Friedman test for all the WFG problems and numbers of objectives. Final ranks, presented in parentheses assigned according to the ranks.

N0	N1	N2	N3	N4	N5	N6	N7
228.0 (4.5)	272.0 (5.0)	**192.0 (3.5)**	262.0 (4.5)	227.0 (4.5)	297.0 (5.0)	223.0 (4.5)	243.0 (4.5)

7.4.3.4 Comparing MOEA/DD-CMA to the Pareto MO-CMA-ES approaches

In this section we compare the Pareto based MO-CMA-ES variants previously proposed with MOEA/DD-CMA. In this comparison, the number of objectives scales from two to fifteen, this is because we hypothesize that that the Pareto based versions perform better for few objectives, but have their performance deteriorated as the number of objectives increases.

The results evaluated by IGD_p and hypervolume are presented in Tables 7.46 and 7.48 respectively. A summary of the results presented in those tables are presented in Tables 7.47 and 7.49 respectively.

The IGD_p results show that for two objectives, the MO-CMA-ES variants outperform MOEA/DD-CMA in all the problems investigated. However, as the number of objectives increases, we can see an inverted pattern, where for three objectives the MO-CMA-ES variants still outperform MOEA/DD-CMA in most problems, but for five objectives onwards, MOEA/DD-CMA outperforms the Pareto based variants in most of the problems, achieving the best results in all problems for ten and fifteen objectives, although without statistically significant difference on problem DTLZ6 for fifteen objectives.

Table 7.46: Mean ranks of the IGD_p as used in the Kruskal-Wallis test for the DTLZ problems. Final ranks, presented in parentheses assigned according to mean ranks.

Obj.	Algorithms	DTLZ1	DTLZ2	DTLZ3	DTLZ4	DTLZ5	DTLZ6	DTLZ7
2	MOEA/DD-CMA	74.47 (3.00)	75.50 (3.00)	73.53 (3.00)	65.70 (3.00)	75.50 (3.00)	64.90 (2.50)	75.20 (3.00)
	MO-CMA-ES$_{CD}$-rankMu	42.37 (2.00)	**15.50 (1.00)**	**30.70 (1.50)**	**35.80 (1.50)**	**15.50 (1.00)**	**19.37 (1.00)**	**15.50 (1.00)**
	MO-CMA-ES$_{CD}$-rankOne	**19.67 (1.00)**	45.50 (2.00)	**32.27 (1.50)**	**35.00 (1.50)**	45.50 (2.00)	52.23 (2.50)	45.80 (2.00)
3	MOEA/DD-CMA	70.20 (3.00)	**15.53 (1.00)**	68.13 (3.00)	**36.23 (1.50)**	75.50 (3.00)	70.57 (3.00)	75.13 (3.00)
	MO-CMA-ES$_{CD}$-rankMu	**33.13 (1.50)**	45.47 (2.00)	45.47 (2.00)	69.70 (3.00)	**15.50 (1.00)**	**15.50 (1.00)**	**20.33 (1.00)**
	MO-CMA-ES$_{CD}$-rankOne	**33.17 (1.50)**	75.50 (3.00)	**22.90 (1.00)**	**30.57 (1.50)**	45.50 (2.00)	50.43 (2.00)	41.03 (2.00)
5	MOEA/DD-CMA	**24.13 (1.00)**	**15.50 (1.00)**	46.00 (2.00)	**16.40 (1.00)**	**15.77 (1.00)**	65.60 (2.50)	75.50 (3.00)
	MO-CMA-ES$_{CD}$-rankMu	54.90 (2.50)	45.50 (2.00)	57.80 (2.50)	44.60 (2.00)	57.83 (2.50)	**19.57 (1.00)**	43.67 (2.00)
	MO-CMA-ES$_{CD}$-rankOne	57.47 (2.50)	75.50 (3.00)	**32.70 (1.50)**	75.50 (3.00)	62.90 (2.50)	51.33 (2.50)	**17.33 (1.00)**
8	MOEA/DD-CMA	**38.10 (2.00)**	**15.50 (1.00)**	**15.60 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**26.25 (1.50)**	75.50 (3.00)
	MO-CMA-ES$_{CD}$-rankMu	**53.80 (2.00)**	45.50 (2.00)	75.50 (3.00)	47.77 (2.00)	52.27 (2.00)	73.98 (3.00)	40.90 (2.00)
	MO-CMA-ES$_{CD}$-rankOne	**44.60 (2.00)**	75.50 (3.00)	45.40 (2.00)	73.23 (3.00)	68.73 (3.00)	**36.27 (1.50)**	**20.10 (1.00)**
10	MOEA/DD-CMA	**20.37 (1.00)**	**15.50 (1.00)**	**16.50 (1.00)**	**15.50 (1.00)**	**15.77 (1.00)**	**18.37 (1.00)**	**22.30 (1.00)**
	MO-CMA-ES$_{CD}$-rankMu	67.47 (3.00)	45.50 (2.00)	75.50 (3.00)	46.23 (2.00)	59.23 (2.50)	73.83 (3.00)	54.63 (2.50)
	MO-CMA-ES$_{CD}$-rankOne	48.67 (2.00)	75.50 (3.00)	44.50 (2.00)	74.77 (3.00)	61.50 (2.50)	44.30 (2.00)	59.57 (2.50)
15	MOEA/DD-CMA	**27.80 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**26.03 (1.50)**	**15.50 (1.00)**
	MO-CMA-ES$_{CD}$-rankMu	59.97 (2.50)	51.23 (2.00)	75.50 (3.00)	45.50 (2.00)	55.20 (2.50)	72.70 (3.00)	65.47 (2.50)
	MO-CMA-ES$_{CD}$-rankOne	48.73 (2.50)	69.77 (3.00)	45.50 (2.00)	75.50 (3.00)	65.80 (2.50)	**37.77 (1.50)**	55.53 (2.50)

Table 7.47: Overall ranks of IGD_p as used in the Friedman test for all the DTLZ problems and numbers of objectives. Final ranks, presented in parentheses assigned according to the ranks.

MOEA/DD-CMA	MO-CMA-ES$_{CD}$-rankMu	MO-CMA-ES$_{CD}$-rankOne
73.0 (2.0)	**89.0 (2.0)**	**90.0 (2.0)**

A similar pattern arises when considering the hypervolume indicator, where for two objectives the MOEA/DD-CMA performs poorly compared to the MO-CMA-ES variants. For three objectives this difference becomes smaller, and from five objectives onwards, MOEA/DD-CMA outperforms the Pareto based variants in most of the problems.

The summarized analysis considering both indicators was not able to identify statistically significant differences between the algorithms, this is mostly because each one performs well in a specific scenario, hence when we erase these different scenarios, the algorithms become competitive.

In an overview of the results, we can confirm our hypothesis on this section that the MO-CMA-ES variants perform well on problems of two and three objectives, however as the

number of objectives increases, their performance deteriorates, as is usual in Pareto based algorithms. In the many-objective scenarios, MOEA/DD-CMA has shown its superiority in most of the problems investigated.

Table 7.48: Mean ranks of the hypervolume as used in the Kruskal-Wallis test for the DTLZ problems. Final ranks, presented in parentheses assigned according to mean ranks.

Obj.	Algorithms	DTLZ1	DTLZ2	DTLZ3	DTLZ4	DTLZ5	DTLZ6	DTLZ7
2	MOEA/DD-CMA	75.00 (3.00)	75.50 (3.00)	72.57 (3.00)	66.60 (3.00)	75.50 (3.00)	66.20 (2.50)	75.50 (3.00)
	MO-CMA-ES$_{CD}$-rankMu	45.70 (2.00)	**15.50 (1.00)**	40.50 (2.00)	**33.90 (1.50)**	**15.50 (1.00)**	**19.03 (1.00)**	**15.50 (1.00)**
	MO-CMA-ES$_{CD}$-rankOne	**15.80 (1.00)**	45.50 (2.00)	**23.43 (1.00)**	**36.00 (1.50)**	45.50 (2.00)	51.27 (2.50)	45.50 (2.00)
3	MOEA/DD-CMA	72.83 (3.00)	45.23 (2.00)	68.90 (3.00)	**35.63 (1.50)**	75.50 (3.00)	74.30 (3.00)	75.50 (3.00)
	MO-CMA-ES$_{CD}$-rankMu	39.90 (2.00)	**15.80 (1.00)**	50.03 (2.00)	66.60 (3.00)	**15.50 (1.00)**	**15.50 (1.00)**	**15.50 (1.00)**
	MO-CMA-ES$_{CD}$-rankOne	**23.77 (1.00)**	75.47 (3.00)	**17.57 (1.00)**	**34.27 (1.50)**	45.50 (2.00)	46.70 (2.00)	45.50 (2.00)
5	MOEA/DD-CMA	**30.40 (1.50)**	**15.50 (1.00)**	71.37 (3.00)	**15.50 (1.00)**	**15.50 (1.00)**	75.50 (3.00)	75.50 (3.00)
	MO-CMA-ES$_{CD}$-rankMu	45.73 (2.00)	45.53 (2.00)	**38.57 (1.50)**	45.50 (2.00)	54.07 (2.50)	**30.63 (1.50)**	**15.50 (1.00)**
	MO-CMA-ES$_{CD}$-rankOne	60.37 (2.50)	75.47 (3.00)	**26.57 (1.50)**	75.50 (3.00)	66.93 (2.50)	**30.37 (1.50)**	45.50 (2.00)
8	MOEA/DD-CMA	75.00 (3.00)	**15.50 (1.00)**	43.43 (2.00)	**15.50 (1.00)**	**15.50 (1.00)**	**36.63 (1.50)**	74.97 (3.00)
	MO-CMA-ES$_{CD}$-rankMu	**21.40 (1.00)**	45.50 (2.00)	71.20 (3.00)	48.50 (2.00)	49.97 (2.00)	75.50 (3.00)	45.17 (2.00)
	MO-CMA-ES$_{CD}$-rankOne	40.10 (2.00)	75.50 (3.00)	**21.87 (1.00)**	72.50 (3.00)	71.03 (3.00)	**24.37 (1.50)**	**16.37 (1.00)**
10	MOEA/DD-CMA	75.50 (3.00)	**15.90 (1.00)**	55.33 (2.50)	**15.50 (1.00)**	**15.50 (1.00)**	**37.43 (1.50)**	73.90 (3.00)
	MO-CMA-ES$_{CD}$-rankMu	**28.47 (1.50)**	45.10 (2.00)	65.67 (2.50)	47.03 (2.00)	52.33 (2.00)	75.50 (3.00)	46.53 (2.00)
	MO-CMA-ES$_{CD}$-rankOne	**32.53 (1.50)**	75.50 (3.00)	**15.50 (1.00)**	73.97 (3.00)	68.67 (3.00)	**23.57 (1.50)**	**16.07 (1.00)**
15	MOEA/DD-CMA	**45.50 (2.00)**	**15.50 (1.00)**	**45.50 (2.00)**	**15.50 (1.00)**	**15.50 (1.00)**	**33.70 (1.50)**	**45.50 (2.00)**
	MO-CMA-ES$_{CD}$-rankMu	**45.50 (2.00)**	59.50 (2.50)	**45.50 (2.00)**	53.87 (2.50)	56.42 (2.50)	70.47 (3.00)	**45.50 (2.00)**
	MO-CMA-ES$_{CD}$-rankOne	**45.50 (2.00)**	61.50 (2.50)	**45.50 (2.00)**	67.13 (2.50)	64.58 (2.50)	**32.33 (1.50)**	**45.50 (2.00)**

Table 7.49: Overall ranks of hypervolume as used in the Friedman test for all the DTLZ problems and numbers of objectives. Final ranks, presented in parentheses assigned according to the ranks.

MOEA/DD-CMA	MO-CMA-ES$_{CD}$-rankMu	MO-CMA-ES$_{CD}$-rankOne
87.0 (2.0)	**79.0 (2.0)**	**86.0 (2.0)**

7.4.3.5 Summary of the results

This section presented MOEA/DD-CMA, a new EDA inspired by MOEA/D-CMA and based on CMA-ES and MOEA/DD. MOEA/DD-CMA uses the selection mechanism of MOEA/DD that employs dominance and decomposition simultaneously to increase the diversity of solutions found.

Three experimental studies were carried out to evaluate the performance of MOEA/DD-CMA on a comprehensive set of many-objective benchmark problems as the number of objectives scales from two to fifteen. The results obtained in these studies were evaluated using two well known quality indicators: IGD_p and Hypervolume. The indicator results calculated over 30 independent runs of the algorithms were submitted to the Kruskal-Wallis statistical test to detect significant differences between the results of the algorithms. Moreover, overall analyses of the results disregarding problem and objective number were provided using the Friedman statistical test.

In the first study we compared the MOEA/DD-CMA with MOEA/D-CMA, which is an algorithm from the literature that combines the classic MOEA/D with the CMA-ES. The results indicate that MOEA/DD-CMA is competitive to or better than MOEA/D-CMA in all of the thirteen problems tested for two and three objectives. Moreover, MOEA/DD-CMA outperformed MOEA/D-CMA in most of the problems from five to fifteen objectives.

In the second study, we compared several neighboring relation approaches to replace the original initialization of the neighborhood of MOEA/D. We found that this approach can be highly dependent on the problem being optimized, moreover, for the DTLZ problems, the

original approach performs well, along to just using the Euclidean distance between the best solutions found by each subproblem. For the WFG problems, it is best to use the likelihood given by a model to the best solution of their neighbors.

Our third study was carried to compare the performances of the Pareto based MO-CMA-ES variants previously proposed in this chapter to MOEA/DD-CMA. In this study we were more interested in the scalability of the algorithms along the number of objectives, and how this increase could affect the results. As expected, the Pareto based approaches performed very well for two and three objectives, however when the number of objectives increased, they performance deteriorated, as is usual in Pareto based approaches. In this scenario, MOEA/DD-CMA outperformed the MO-CMA-ES variants in most of the cases investigated.

7.5 Discussion

In this chapter we proposed three very different approaches for employing the CMA-ES models in the optimization of multiple objectives. The first of these approaches used a single CMA-ES model, and several methods for injecting information in this model were proposed and called TWFs.

The second approach employs a Pareto based framework, where each individual of the population has its own CMA-ES model. This model can be updated using two different variants. The first of them uses information only from the current solution and its offspring to update its model. The second model update method uses the information of the T closest individuals to update the model. Moreover, two variants of the framework using each model update mechanism were proposed. One of them uses an unbounded population, where all non-dominated solutions found during the search are kept, the other uses a crowding distance archiver to ensure an upper limit of the population.

The third approach employs a dominance and decomposition based framework, where each subproblem has its own CMA-ES model. This model is updated based on the information of its neighborhood, as in the classical MOEA/D framework. Eight different approaches for calculating, or recalculating the neighborhood at each iteration, were presented.

A comprehensive set of empirical studies was presented to identify the best algorithm variants or components. Moreover, studies were conducted to compare the proposed algorithms to classical MOEAs from the literature and variants of them employing the CMA-ES model. Finally our best algorithms were compared to identify the best one under multi and many objective scenarios.

The results indicated that, for the hard COCO family of benchmark problems, even the least powerful single model MO-CMA-ES variants are able to outperform the classical MOEAs in most of the scenarios studied. However, when using the classical DTLZ problems, the multiple model MO-CMA-ES algorithms performed better.

As previously hypothesized, the Pareto based MO-CMA-ES variants, MO-CMA-ES-rankOne and MO-CMA-ES-rankMu outperformed the dominance and decomposition based version MOEA/DD-CMA for two and three objectives. However, in the experiments where a higher number of objectives was considered, MOEA/DD-CMA has shown its superiority, outperforming MO-CMA-ES-rankOne and MO-CMA-ES-rankMu in most of the problems.

Future works include investigating in detail the behavior of MO-CMA-ES for problems where it does not perform well, like DTLZ1 and DTLZ3. In order to aid in this investigation, other works should measure the quality of the CMA-ES models as the search progresses to track the progress (or not) of the models.

Chapter 8

Conclusion

In this work, we investigated alternatives to devise bio-inspired optimizers able to deal with MOPs and MaOPs. This investigation leads us to propose approaches which integrate three very different research lines. These research lines represent novel, state-of-the-art approaches in the bio-inspired optimization field.

The first of these research lines involves seeking alternatives to enhance the search ability of the MOPSO algorithm. Our research group is investigating MOPSO for several years and has acquired knowledge on the working principles of this algorithm. Previous researches (Britto and Pozo, 2012; Castro Jr. et al., 2012) in our group have indicated that the leader and archiving methods can positively influence the performance of a MOPSO algorithm. Hence, in this work, we proposed to merge the MOPSO framework with the hyper-heuristic approach, which has been gaining increasing attention in the community. As a result, we presented the H-MOPSO framework, which employs hyper-heuristics to dynamically select leader and archiving methods for a MOPSO. Several empirical studies were conducted to investigate its performance and to parameterize H-MOPSO. Moreover, an empirical study was conducted to validate the performance of H-MOPSO by comparing it to the state-of-the-art MOEA/D-FRRMAB. The results of these experiments indicate that the H-MOPSO framework can achieve excellent results in terms of generated solutions, and even outperform a state-of-the-art algorithm in most of the investigated problems.

Our second research line involves investigating alternatives to enhance the performance of the I-Multi MOPSO, recently proposed by our research group. This algorithm employs multiple swarms, therefore, each sub-swarm can concentrate on a smaller part of the search space and achieve higher convergence, while the well spread of these sub-swarms increases the diversity. In order to explore this second research line, we propose two different approaches: one of them investigates the influence of the clustering strategy in the inner working of the algorithm. The other investigates the hybridization of MOPSO with an EDA approach to increase the convergence of each sub-swarm.

In a first study involving I-Multi, we investigated two important characteristics of its clustering strategy. The characteristics were the space in which the clustering is performed and the similarity metric used to compare the solutions. As result of the conducted experimental studies, we can recommend clustering the solutions in the objective space, since it presented good results in most of the problems. Considering the clustering metrics, the results indicate that, in general, the algorithm is not sensitive to the choice of the metric, however, this sensitivity increases with the number of objectives. This choice can also be problem-dependent, but in overall, the original Euclidean distance can be recommended.

In a second study involving I-Multi, we combined it with the EDA rBOA, resulting in the new C-Multi algorithm. Empirical studies were conducted to parameterize C-Multi, as well as to validate its performance by comparing it to the state-of-the-art MOEA/D-DRA, winner of the CEC09 contest, and to its base algorithm I-Multi. The obtained results indicate that despite not being able to outperform MOEA/D-DRA in most of the cases regarding hypervolume, C-Multi had very good results when considering the IGD_p indicator. Moreover, C-Multi was able to outperform I-Multi in most of the cases regarding IGD_p and presented competitive results according to hypervolume.

Our third research line was motivated by the knowledge gathered during the previous work on EDAs and by the partnership with the University of the Basque Country (UPV/EHU). In order to devise good multi-objective EDAs, we selected a state-of-the-art variant called CMA-ES and proposed three very different approaches based on it. The first of them was the least powerful and used a single CMA-ES model. The second approach employed a Pareto based framework, here each individual of the population has its own CMA-ES model. The third approach employs a dominance and decomposition based framework, where each subproblem has its own CMA-ES model. The results of the conducted empirical studies indicated that for the hard COCO family of benchmark problems, even the least powerful single model MO-CMA-ES variants were able to outperform the classical MOEAs in most of the scenarios studied. However, when optimizing the classical DTLZ problems, this single model MO-CMA-ES was outperformed by classical MOEAs. The multiple model MO-CMA-ES variants on the other hand were able to outperform the best of the classical MOEAs (IBEA) on the DTLZ problems. In a last experimental study, the multiple model MO-CMA-ES variants was compared with the dominance and decomposition MO-CMA-ES (MOEA/DD-CMA). In this scenario, the MO-CMA-ES variants performed better for two and three objectives, however MOEA/DD-CMA outperformed them when the number of objectives increased.

Future works include comparing, enhancing and combining the different lines of research explored in this work. To combine these lines, a possible alternative is to create a multi-swarm (or multi-population) algorithm where each sub-swarm uses an optimization algorithm with its own parameters (i.e., EDAs or MOPSOs), moreover, hyper-heuristics can be used to dynamically select the best algorithm and/or configuration to each sub-swarm.

References

Adra, S. F. (2007). *Improving Convergence, Diversity and Pertinency in Multiobjective Optimisation*. PhD thesis, Department of Automatic Control and Systems Engineering, The University of Sheffield, UK.

Aggarwal, C. C., Hinneburg, A., and Keim, D. A. (2001). On the surprising behavior of distance metrics in high dimensional space. In *International Conference on Database Theory*, pages 420–434, London, UK.

Ahn, C., Ramakrishna, R. S., and Goldberg, D. E. (2006). Real-coded Bayesian optimization algorithm. In *Towards a New Evolutionary Computation*, volume 192 of *Studies in Fuzziness and Soft Computing*, pages 51–73. Springer Heidelberg Germany.

Ahn, C. W. (2006). *Advances in Evolutionary Algorithms: Theory, Design and Practice*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Ahn, C. W. and Ramakrishna, R. S. (2008). On the scalability of real-coded Bayesian optimization algorithm. *IEEE Transactions on Evolutionary Computation*, 12(3):307–322.

Asafuddoula, M., Ray, T., and Sarker, R. (2015). A decomposition-based evolutionary algorithm for many objective optimization. *IEEE Transactions on Evolutionary Computation*, 19(3):445–460.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2):235–256.

Auger, A. and Hansen, N. (2005). A restart CMA evolution strategy with increasing population size. In *IEEE Congress on Evolutionary Computation*, volume 2, pages 1769–1776, Edinburgh, UK.

Bader, J., Deb, K., and Zitzler, E. (2010). Faster hypervolume-based search using Monte Carlo sampling. In *Multiple Criteria Decision Making for Sustainable Energy and Transportation Systems*, volume 634 of *Lecture Notes in Economics and Mathematical Systems*, pages 313–326. Springer Heidelberg Germany.

Bader, J. and Zitzler, E. (2011). Hype: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary Computation*, 19(1):45–76.

Bansal, J. C. and Deep, K. (2012). A modified binary particle swarm optimization for knapsack problems. *Applied Mathematics and Computation*, 218(22):11042–11061.

Bastos-Filho, C. J. A. and Guimarães, A. C. S. (2015). Multi-objective fish school search. *International Journal of Swarm Intelligence Research*, 6(1):23–40.

Bechikh, S., Elarbi, M., and Ben Said, L. (2017). *Many-objective Optimization Using Evolutionary Algorithms: A Survey*, pages 105–137. Springer International Publishing, Cham.

Beume, N., Naujoks, B., and Emmerich, M. (2007). SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653 – 1669.

Beyer, H. and Schwefel, H. (2002). Evolution strategies – a comprehensive introduction. *Natural Computing*, 1(1):3–52.

Bilgin, B., Özcan, E., and Korkmaz, E. E. (2007). An experimental study on hyper-heuristics and exam timetabling. In *Practice and Theory of Automated Timetabling VI*, volume 3867 of *Lecture Notes in Computer Science*, pages 394–412. Springer Heidelberg Germany.

Bosman, P. A. (2003). *Design and Application of Iterated Density-Estimation Evolutionary Algorithms*. PhD thesis, Universiteit Utrecht, Utrecht, The Netherlands.

Branke, J. and Mostaghim, S. (2006). About selecting the personal best in multi-objective particle swarm optimization. In *Parallel Problem Solving from Nature - PPSN IX*, Lecture Notes in Computer Science, pages 523–532. Springer Berlin / Heidelberg.

Bringmann, K., Friedrich, T., Igel, C., and Voß, T. (2013). Speeding up many-objective optimization by Monte Carlo approximations. *Artificial Intelligence*, 204:22–29.

Britto, A., Mostaghim, S., and Pozo, A. (2013). Iterated multi-swarm: A multi-swarm algorithm based on archiving methods. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, pages 583–590, New York, NY, USA. ACM.

Britto, A. and Pozo, A. (2012). Using archiving methods to control convergence and diversity for many-objective problems in particle swarm optimization. In *IEEE Congress on Evolutionary Computation*, pages 1–8, Brisbane, AU.

Britto, A. and Pozo, A. (2015). Reference-point based multi-swarm algorithm for many-objective problems. In *Brazilian Conference on Intelligent Systems (BRACIS)*, pages 252–257.

Brockhoff, D., Tušar, T., Tušar, D., Wagner, T., Hansen, N., and Auger, A. (2016). Biobjective Performance Assessment with the COCO Platform. *ArXiv e-prints* .

Brockhoff, D., Wagner, T., and Trautmann, H. (2012). On the properties of the R2 indicator. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, pages 465–472, New York, NY, USA. ACM.

Brownlee, A. E. I. and Wright, J. A. (2012). Solution analysis in multi-objective optimization. *Building Simulation and Optimization*, pages 317–324.

Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., and Qu, R. (2013). Hyper-heuristics: a survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724.

Castro Jr., O. R., Britto, A., and Pozo, A. (2012). A comparison of methods for leader selection in many-objective problems. In *IEEE Congress on Evolutionary Computation*, pages 1–8, Brisbane, AU.

Castro Jr., O. R., Fritsche, G. M., and Pozo, A. Evaluating selection methods on hyper-heuristic multi-objective particle swarm optimization. *Journal of Heuristics*. submitted.

Castro Jr., O. R. and Pozo, A. (2014a). A hybrid competent multi-swarm approach for many-objective problems. In *Brazilian Conference on Intelligent Systems*, pages 426–431, Sao Paulo, BR.

Castro Jr., O. R. and Pozo, A. (2014b). A MOPSO based on hyper-heuristic to optimize many-objective problems. In *IEEE Symposium on Swarm Intelligence (SIS)*, pages 1–8, Orlando, FL, USA.

Castro Jr., O. R. and Pozo, A. (2015). Using hyper-heuristic to select leader and archiving methods for many-objective problems. In *Evolutionary Multi-Criterion Optimization*, volume 9018 of *Lecture Notes in Computer Science*, pages 109–123. Springer International Publishing.

Castro Jr., O. R., Pozo, A., Lozano, J. A., and Santana, R. Combining CMA-ES and MOEA/DD for many-objective optimization. In *IEEE Congress on Evolutionary Computation*. submitted.

Castro Jr., O. R., Pozo, A., Lozano, J. A., and Santana, R. An investigation of clustering strategies in many-objective optimization: I-Multi as a case study. *Swarm Intelligence*. submitted.

Castro Jr., O. R., Pozo, A., Lozano, J. A., and Santana, R. (2016a). Transfer weight functions for injecting problem information in the multi-objective CMA-ES. *Memetic Computing*, pages 1–28.

Castro Jr., O. R., Santana, R., and Pozo, A. (2016b). C-Multi: a competent multi-swarm approach for many-objective problems. *Neurocomputing*, 180:68 – 78. Progress in Intelligent Systems Design Selected papers from the 4th Brazilian Conference on Intelligent Systems (BRACIS 2014).

Chen, L. and Liu, H. (2014). A region decomposition-based multi-objective particle swarm optimization algorithm. *International Journal of Pattern Recognition and Artificial Intelligence*, 28(8):1459009.

Clerc, M. and Kennedy, J. (2002). The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73.

Coello, C. A. C. and Cortés, N. C. (2005). Solving multiobjective optimization problems using an artificial immune system. *Genetic Programming and Evolvable Machines*, 6(2):163–190.

Coello, C. A. C., Dehuri, S., and Ghosh, S. (2009). *Swarm Intelligence for Multi-objective Problems in Data Mining*. Springer Publishing Company, Incorporated, 1st edition.

Coello, C. A. C., Lamont, G. B., and Veldhuizen, D. A. V. (2006). *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Corder, G. W. and Foreman, D. I. (2014). *Nonparametric Statistics: A Step-by-Step Approach*. Wiley.

Corne, D. and Knowles, J. (2003). Some multiobjective optimizers are better than others. In *IEEE Congress on Evolutionary Computation*, volume 4, pages 2506–2512.

Cowling, P., Kendall, G., and Soubeiga, E. (2001). A Hyperheuristic Approach to Scheduling a Sales Summit. In Burke, E. and Erben, W., editors, *Practice and Theory of Automated Timetabling III*, volume 2079 of *Lecture Notes in Computer Science*, pages 176–190. Springer Heidelberg Germany.

Dai, C., Wang, Y., and Ye, M. (2015). A new multi-objective particle swarm optimization algorithm based on decomposition. *Information Sciences*, 325:541–557.

Das, I. and Dennis, J. E. (1998). Normal-boundary intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems. *SIAM Journal on Optimization*, 8(3):631–657.

Davies, D. L. and Bouldin, D. W. (1979). A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):224–227.

de Carvalho, V. R. and Pozo, A. P. (2014). Um estudo sobre otimização por partículas aplicado ao problema de roteamento de veículos com demandas estocásticas. In *ENIAC*, pages 1–6, São Carlos, BR.

Deb, K. (2009). *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley.

Deb, K. and Agrawal, R. B. (1995). Simulated binary crossover for continuous search space. *Complex Systems*, 9(2):115–148.

Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation: NSGA-II. In *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, PPSN VI, pages 849–858, London, UK. Springer-Verlag.

Deb, K. and Jain, H. (2014). An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601.

Deb, K., Thiele, L., Laumanns, M., and Zitzler, E. (2002). Scalable multi-objective optimization test problems. In *IEEE Congress on Evolutionary Computation*, volume 1, pages 825–830.

Deb, K., Thiele, L., Laumanns, M., and Zitzler, E. (2005). Scalable test problems for evolutionary multiobjective optimization. In *Evolutionary Multiobjective Optimization*, Advanced Information and Knowledge Processing, pages 105–145. Springer London.

Deborah, H., Richard, N., and Hardeberg, J. Y. (2015). A comprehensive evaluation of spectral distance functions and metrics for hyperspectral image processing. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 8(6):3224–3234.

Demsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30.

Derrac, J., García, S., Molina, D., and Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18.

Deza, M. (2009). *Encyclopedia of distances*. Springer-Verlag, Berlin Heidelberg.

Díaz-Manríquez, A., Toscano, G., Barron-Zambrano, J. H., and Tello-Leal, E. (2016). R2-based multi/many-objective particle swarm optimization. *Computational Intelligence and Neuroscience*, 2016:1–10.

Drake, J. H., Ozcan, E., and Burke, E. K. (2012). An improved choice function heuristic selection for cross domain heuristic search. In *Parallel Problem Solving from Nature - PPSN XII*, volume 7492 of *Lecture Notes in Computer Science*, pages 307–316. Springer Heidelberg Germany.

Durillo, J. J., García-Nieto, J., Nebro, A. J., Coello, C. A. C., Luna, F., and Alba, E. (2009). Multi-objective particle swarm optimizers: An experimental comparison. In *Proceedings of the 5th International Conference on Evolutionary Multi-Criterion Optimization*, EMO '09, pages 495–509, Heidelberg, Germany. Springer-Verlag.

Durillo, J. J., Nebro, A. J., and Alba, E. (2010). The jMetal framework for multi-objective optimization: Design and architecture. In *IEEE Congress on Evolutionary Computation*, pages 4138–4325, Barcelona, ES.

Eberhart, R. C. and Shi, Y. (2001). Particle swarm optimization: developments, applications and resources. In *Proceedings of the 2001 Congress on Evolutionary Computation*, volume 1, pages 81–86, Seoul, KR.

Fialho, A., Da Costa, L., Schoenauer, M., and Sebag, M. (2010). Analyzing bandit-based adaptive operator selection mechanisms. *Annals of Mathematics and Artificial Intelligence*, 60(1-2):25–64.

Fonseca, C. M. and Fleming, P. J. (1998). Multiobjective optimization and multiple constraint handling with evolutionary algorithms - part I: A unified formulation. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 28(1):26–37.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701.

García, I. C., Coello, C. A. C., and Arias-Montaño, A. (2014). MOPSOhv: A new hypervolume-based multi-objective particle swarm optimizer. In *IEEE Congress on Evolutionary Computation (CEC)*, Beijing, CN.

Glover, F. (2003). *Handbook of metaheuristics*. Kluwer Academic Publishers, Boston.

Gonçalves, R. A., Almeida, C. P., and Pozo, A. (2015a). Upper confidence bound (UCB) algorithms for adaptive operator selection in MOEA/D. In *Evolutionary Multi-Criterion Optimization*, volume 9018 of *Lecture Notes in Computer Science*, pages 411–425. Springer International Publishing.

Gonçalves, R. A., Kuk, J. N., Almeida, C. P., and Venske, S. M. (2015b). MOEA/D-HH: A hyper-heuristic for multi-objective problems. In *Evolutionary Multi-Criterion Optimization*, volume 9018 of *Lecture Notes in Computer Science*, pages 94–108. Springer International Publishing.

Guizzo, G., Fritsche, G. M., Vergilio, S. R., and Pozo, A. P. (2015). A hyper-heuristic for the multi-objective integration and test order problem. In *Proceedings of the 2015 Genetic and Evolutionary Computation Conference*, pages 1343–1350, New York, NY, USA. ACM.

Guo, W., Chen, M., Wang, L., and Wu, Q. (2016). Hyper multi-objective evolutionary algorithm for multi-objective optimization problems. *Soft Computing*, pages 1–9.

Hansen, M. P. and Jaszkiewicz, A. (1998). Evaluating the quality of approximations to the non-dominated set. Technical Report IMM-REP-1998-7, Technical University of Denmark.

Hansen, N. (2011). Injecting external solutions into CMA-ES. *CoRR*, abs/1110.4181.

Hansen, N., Auger, A., Mersmann, O., Tusar, T., and Brockhoff, D. (2016). COCO: A platform for comparing continuous optimizers in a black-box setting. *CoRR*, abs/1603.08785.

Hansen, N. and Ostermeier, A. (1996). Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In *IEEE International Conference on Evolutionary Computation*, pages 312–317, Nagoya, JP.

Harik, G., Cantú-Paz, E., Goldberg, D. E., and Miller, B. L. (1999). The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Evolutionary Computation*, 7(3):231–253.

Hartigan, J. A. and Wong, M. A. (1979). Algorithm AS 136: A K-means clustering algorithm. *Applied Statistics*, 28(1):100–108.

Hauschild, M. and Pelikan, M. (2011). An introduction and survey of estimation of distribution algorithms. *Swarm and Evolutionary Computation*, 1(3):111–128.

Henrion, M. (1986). Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In *2 Annual Conference on Uncertainty in Artificial Intelligence (UAI-86)*, pages 149–163, Amsterdam, NL. Elsevier Science.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press.

Howarth, P. and Rüger, S. (2005). Fractional distance measures for content-based image retrieval. In *European Conference on Information Retrieval*, pages 447–456. Springer.

Huband, S., Hingston, P., Barone, L., and While, L. (2006). A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506.

Igel, C., Hansen, N., and Roth, S. (2007a). Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation*, 15(1):1–28.

Igel, C., Suttorp, T., and Hansen, N. (2007b). Steady-state selection and efficient covariance matrix update in the multi-objective CMA-ES. In Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., and Murata, T., editors, *Evolutionary Multi-Criterion Optimization*, volume 4403 of *Lecture Notes in Computer Science*, pages 171–185. Springer Heidelberg Germany.

Ishibuchi, H., Akedo, N., Ohyanagi, H., and Nojima, Y. (2011). Behavior of EMO algorithms on many-objective optimization problems with correlated objectives. In *IEEE Congress of Evolutionary Computation (CEC)*, pages 1465–1472, New Orleans, LA, USA.

Ishibuchi, H., Tsukamoto, N., and Nojima, Y. (2008). Evolutionary many-objective optimization: A short review. In *IEEE Congress on Evolutionary Computation*, pages 2419–2426.

Jaimes, A. L., Oyama, A., and Fujii, K. (2013). Space trajectory design: Analysis of a real-world many-objective optimization problem. In *IEEE Congress on Evolutionary Computation*, pages 2809–2816, Cancun, MX.

Kaufman, L. and Rousseeuw, P. (1987). Clustering by means of medoids. *Statistical Data Analysis Based on the L1–Norm and Related Methods*.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, volume 4, pages 1942–948.

Kheiri, A., Özcan, E., and Parkes, A. J. (2016). A stochastic local search algorithm with adaptive acceptance for high-school timetabling. *Annals of Operations Research*, 239(1):135–151.

Krause, O., Glasmachers, T., Hansen, N., and Igel, C. (2016). Unbounded population MO-CMA-ES for the bi-objective BBOB test suite. In *Proceedings of the 2016 Genetic and Evolutionary Computation Conference*, pages 1177–1184, New York, NY, USA. ACM.

Kruskal, W. H. and Wallis, W. A. (1952). Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260):583–621.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86.

Landa, R., Coello, C. A. C., and Toscano-Pulido, G. (2013). Goal-constraint: Incorporating preferences through an evolutionary ϵ-constraint based method. In *2013 IEEE Congress on Evolutionary Computation*, pages 741–747.

Larrañaga, P. and Lozano, J. A., editors (2002). *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, Boston/Dordrecht/London.

Larrañaga, P., Karshenas, H., Bielza, C., and Santana, R. (2012). A review on probabilistic graphical models in evolutionary computation. *Journal of Heuristics*, 18(5):795–819.

Laumanns, M. and Zenklusen, R. (2011). Stochastic convergence of random search methods to fixed size Pareto front approximations. *European Journal of Operational Research*, 213(2):414–421.

Li, H. and Zhang, Q. (2009). Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II. *IEEE Transactions on Evolutionary Computation*, 13(2):284–302.

Li, K., Deb, K., Zhang, Q., and Kwong, S. (2015). An evolutionary many-objective optimization algorithm based on dominance and decomposition. *IEEE Transactions on Evolutionary Computation*, 19(5):694–716.

Li, K., Fialho, A., Kwong, S., and Zhang, Q. (2014). Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 18(1):114–130.

Lin, T., Zhang, H., Zhang, K., Tu, Z., and Cui, N. (2016). An adaptive multiobjective estimation of distribution algorithm with a novel Gaussian sampling strategy. *Soft Computing*, pages 1–19.

Liu, H. L., Gu, F., and Zhang, Q. (2014). Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems. *IEEE Transactions on Evolutionary Computation*, 18(3):450–455.

Loshchilov, I. (2013). CMA-ES with restarts for solving CEC 2013 benchmark problems. In *IEEE Congress on Evolutionary Computation*, pages 369–376, Cancun, MX.

Lozano, J. A. and Larrañaga, P. (1999). Applying genetic algorithms to search for the best hierarchical clustering of a dataset. *Pattern Recognition Letters*, 20(9):911–918.

Luke, S. (2013). *Essentials of Metaheuristics*. Lulu, second edition. Available for free at http://cs.gmu.edu/∼sean/book/metaheuristics/.

Luo, J., Liu, Q., Yang, Y., Li, X., rong Chen, M., and Cao, W. (2017). An artificial bee colony algorithm for multi-objective optimisation. *Applied Soft Computing*, 50:235 – 251.

López-Ibáñez, M., Knowles, J., and Laumanns, M. (2011). On sequential online archiving of objective vectors. In *Evolutionary Multi-Criterion Optimization*, volume 6576 of *Lecture Notes in Computer Science*, pages 46–60. Springer Heidelberg Germany.

López-Jaimes, A. and Coello, C. A. C. (2014). Including preferences into a multiobjective evolutionary algorithm to deal with many-objective engineering optimization problems. *Information Sciences*, 277:1–20.

Maashi, M., Özcan, E., and Kendall, G. (2014). A multi-objective hyper-heuristic based on choice function. *Expert Systems with Applications*, 41(9):4475–4493.

Maltese, J., Ombuki-Berman, B., and Engelbrecht, A. (2015). Co-operative vector-evaluated particle swarm optimization for multi-objective optimization. In *2015 IEEE Symposium Series on Computational Intelligence*, pages 1294–1301.

Martí, L., García, J., Berlanga, A., and Molina, J. M. (2016). MONEDA: scalable multi-objective optimization with a neural network-based estimation of distribution algorithm. *Journal of Global Optimization*, 66(4):729–768.

McLachlan, G. and Peel, D. (2000). *Finite Mixture Models*. Wiley-Interscience.

Mostaghim, S. and Teich, J. (2003). Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO). In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, pages 26–33.

Nebro, A. J., Durillo, J. J., Garcia-Nieto, J., Coello, C. A. C., Luna, F., and Alba, E. (2009). SMPSO: A new PSO-based metaheuristic for multi-objective optimization. In *Computational intelligence in multi-criteria decision-making*, pages 66–73. IEEE.

Nemenyi, P. (1963). *Distribution-free Multiple Comparisons*. PhD thesis, Princeton University.

Niu, B., Wang, H., Wang, J., and Tan, L. (2013). Multi-objective bacterial foraging optimization. *Neurocomputing*, 116:336 – 345. Advanced Theory and Methodology in Intelligent Computing Selected Papers from the Seventh International Conference on Intelligent Computing (ICIC 2011).

Ozcan, E., Bykov, Y., Birben, M., and Burke, E. K. (2009). Examination timetabling using late acceptance hyper-heuristics. In *IEEE Congress on Evolutionary Computation*, pages 997–1004, Trondheim, NO.

Padhye, N., Branke, J., and Mostaghim, S. (2009). Empirical comparison of MOPSO methods: guide selection and diversity preservation. In *Proceedings of the Eleventh Congress on Evolutionary Computation*, pages 2516–2523, Trondheim, NO. IEEE Press.

Parsopoulos, K. E. and Vrahatis, M. N. (2002). Particle swarm optimization method in multiobjective problems. In *Proceedings of the 2002 ACM Symposium on Applied Computing*, SAC '02, pages 603–607, New York, NY, USA. ACM.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Pelikan, M. (2002). *Bayesian optimization algorithm: From single level to hierarchy.* PhD thesis, University of Illinois.

Pelikan, M. (2005). *Hierarchical Bayesian optimization algorithm toward a new generation of evolutionary algorithms*. Springer-Verlag, Berlin, Germany.

Pelikan, M., Goldberg, D. E., and Cantu-Paz, E. (1999). BOA: The Bayesian Optimization Algorithm. pages 525–532. Morgan Kaufmann.

Pelikan, M., Sastry, K., and Cantú-Paz, E. (2006). *Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications (Studies in Computational Intelligence)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Phan, D. H. and Suzuki, J. (2013). R2-IBEA: R2 indicator based evolutionary algorithm for multiobjective optimization. In *IEEE Congress on Evolutionary Computation*, pages 1836–1845, Cancun, MX.

Preuss, M., Kausch, C., Bouvy, C., and Henrich, F. (2010). *Decision Space Diversity Can Be Essential for Solving Multiobjective Real-World Problems*, pages 367–377. Springer, Heidelberg, Germany.

Reyes-Sierra, M. and Coello, C. A. C. (2006). Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *International Journal of Computational Intelligence Research*, 2(3):287–308.

Russell, S. and Norvig, P. (2011). *Artificial Intelligence: A Modern Approach*. Prentice Hall.

Salcedo-Sanz, S., Pastor-Sánchez, A., Portilla-Figueras, J. A., and Prieto, L. (2015). Effective multi-objective optimization with the coral reefs optimization algorithm. *Engineering Optimization*, 48(6):966–984.

Santana, R., Larrañaga, P., and Lozano, J. A. (2009). Research topics on discrete estimation of distribution algorithms. *Memetic Computing*, 1(1):35–54.

Santana, R., Larrañaga, P., and Lozano, J. A. (2010). Learning factorizations in estimation of distribution algorithms using affinity propagation. *Evolutionary Computation*, 18(4):515–546.

Saxena, D. K., Duro, J. A., Tiwari, A., Deb, K., and Zhang, Q. (2013). Objective reduction in many-objective optimization: Linear and nonlinear algorithms. *IEEE Transactions on Evolutionary Computation*, 17(1):77–99.

Scheepers, C. and Engelbrecht, A. P. (2016). Vector evaluated particle swarm optimization exploration behavior part I: Explorative analysis. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 1845–1854, Vancouver, CA.

Schutze, O., Esquivel, X., Lara, A., and Coello, C. A. C. (2012). Using the averaged Hausdorff distance as a performance measure in evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 16(4):504–522.

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464.

Shakya, S. and McCall, J. (2007). Optimization by estimation of distribution with DEUM framework based on Markov random fields. *International Journal of Automation and Computing*, 4(3):262–272.

Shi, Y. and Eberhart, R. (1998). A modified particle swarm optimizer. In *IEEE International Conference on Evolutionary Computation*, pages 69–73, Anchorage, AK, USA.

Sierra, M. R. and Coello, C. A. C. (2005). Improving PSO-based multi-objective optimization using crowding, mutation and ϵ-dominance. In *Evolutionary Multi-Criterion Optimization*, volume 3410 of *Lecture Notes in Computer Science*, pages 505–519. Springer Heidelberg Germany.

Suresh, S., Sujit, P. B., and Rao, A. K. (2007). Particle swarm optimization approach for multi-objective composite box-beam design. *Composite Structures*, 81(4):598 – 605.

Tomita, K., Miyakawa, M., and Sato, H. (2015). Adaptive control of dominance area of solutions in evolutionary many-objective optimization. *New Mathematics and Natural Computation*, 11(2):135–150.

Valdez-Peña, I. S., Hernández-Aguirre, A., and Botello-Rionda, S. (2009). Approximating the search distribution to the selection distribution in EDAs. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2009*, pages 461–468, New York, NY, USA. ACM.

van den Bergh, F., Engelbrecht, A. P., and Engelbrecht, A. P. (2000). Cooperative learning in neural networks using particle swarm optimizers. *South African Computer Journal*, 26:84–90.

Veldhuizen, D. A. V. and Lamont, G. B. (1998). Evolutionary computation and convergence to a Pareto front. In *Stanford University, California*, pages 221–228. Morgan Kaufmann.

von Lücken, C., Monzón, H., Brizuela, C., and Barán, B. (2015). Dimensionality reduction in many-objective problems combining PCA and spectral clustering. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO Companion '15, pages 1511–1512, New York, NY, USA. ACM.

von Lücken, C., Barán, B., and Brizuela, C. (2014). A survey on multi-objective evolutionary algorithms for many-objective problems. *Computational Optimization and Applications*, 58(3):707–756.

Voß, T., Beume, N., Rudolph, G., and Igel, C. (2008). Scalarization versus indicator-based selection in multi-objective CMA evolution strategies. In *Proceedings of the 2008 Congress on Evolutionary Computation CEC-2008*, pages 3036–3043, Hong Kong, HK.

Voß, T., Hansen, N., and Igel, C. (2010). Improved step size adaptation for the MO-CMA-ES. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, pages 487–494, New York, NY, USA. ACM.

Vrugt, J. A. and Robinson, B. A. (2007). Improved evolutionary optimization from genetically adaptive multimethod search. *Proceedings of the National Academy of Sciences of the United States of America*, 104(3):708–711. cited By 225.

Wang, Y. and Xu, H. (2014). Multiobjective particle swarm optimization without the personal best. *Journal of Shanghai Jiaotong University (Science)*, 19(2):155–159.

While, L., Bradstreet, L., and Barone, L. (2012). A fast way of calculating exact hypervolumes. *IEEE Transactions on Evolutionary Computation*, 16(1):86–95.

Wilcoxon, F. (1945). Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6):80–83.

Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82.

Zangari, M., Mendiburu, A., Santana, R., and Pozo, A. (2017a). Multiobjective decomposition-based mallows models estimation of distribution algorithm. a case of study for permutation flowshop scheduling problem. *Information Sciences*, 397-398:137–154.

Zangari, M., Santana, R., Mendiburu, A., and Pozo, A. (2017b). Not all {PBILs} are the same: Unveiling the different learning mechanisms of {PBIL} variants. *Applied Soft Computing*, 53:88–96.

Zapotecas-Martínez, S., Derbel, B., Liefooghe, A., Brockhoff, D., Aguirre, H. E., and Tanaka, K. (2015). Injecting CMA-ES into MOEA/D. In *Proceedings of the 2015 Genetic and Evolutionary Computation Conference*, pages 783–790, Madrid, ES. ACM.

Zhang, Q. and Li, H. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731.

Zhang, Q., Liu, W., and Li, H. (2009). The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances. In *IEEE Congress on Evolutionary Computation*, pages 203–208, Trondheim, NO.

Zhang, X., Tian, Y., and Jin, Y. (2015). A knee point-driven evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 19(6):761–776.

Zhu, C., Xu, L., and Goodman, E. D. (2016). Generalization of Pareto-optimality for many-objective evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 20(2):299–315.

Zitzler, E., Deb, K., and Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195.

Zitzler, E. and Künzli, S. (2004). *Indicator-Based Selection in Multiobjective Search*, pages 832–842. Springer, Birmingham, UK.

Zitzler, E., Laumanns, M., and Thiele, L. (2002). SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In *Evolutionary Methods for Design, Optimisation, and Control*, pages 95–100. CIMNE, Barcelona, ES.

Zitzler, E. and Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271.